МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок проходження онлайн-тестів

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21910804

Здобувач __ _____Є. В. Біба

підпис

«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____ Г. В. Горбань

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

м. Миколаїв – 2024 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано здобувач групи 409 факультету комп'ютерних наук

_____Біба Євгеній Валерійович_____

(прізвище, ім'я, по батькові здобувач)

1. Тема кваліфікаційної роботи

Вебзастосунок проходження онлайн-тестів

_____

_____

Затверджена наказом по ЧНУ від «22» ___грудня_ _2023 р. № ___269___

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Вебзастосунок проходження онлайн-тестів

_____

_____

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування програмного забезпечення;
- моделювання та проєктування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення;

5. Перелік графічних матеріалів

__Презентація_____

_

6. Завдання до спеціальної частини

__Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи _____канд. техн. наук, доцент Горбань Г. В._____

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Біба Євгеній Валерійович_____

(прізвище, ім’я, по батькові)

_____

(підпис)

Дата видачі завдання «____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: «Вебзастосунок проходження онлайн-тестів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	20.02.2024	23.02.2024	Виконано
2	Огляд літератури за темою роботи	24.02.2024	26.02.2024	Виконано
3	Складання календарного плану КРБ	26.02.2024	27.02.2024	Виконано
4	Аналіз предметної області	28.02.2024	01.03.2024	Виконано
5	Розробка проєктних рішень	04.03.2024	10.03.2024	Виконано
6	Моделювання та конструювання ПЗ	22.03.2024	26.03.2024	Виконано
7	Кодування ПЗ, тестування та апробація розробленого ПЗ, аналіз результатів тестування	28.03.2024	22.05.2024	Виконано
8	Розробка спеціальної частини з охорони праці	22.03.2024	28.03.2024	Виконано
9	Оформлення КРБ та презентації	22.03.2024	28.03.2024	Виконано
10	Відгук керівника КРБ	02.06.2024	02.06.2024	Виконано
11	Попередній захист	04.06.2024	04.06.2024	Виконано
12	Рецензування	13.06.2024	14.06.2024	Виконано
13	Захист кваліфікаційної роботи	25.06.2024	26.06.2024	Виконано

Розробив здобувач Біба Євгеній Валерійович _____
(прізвище, ім'я, по батькові) *(підпис)*
«24» січня 2024 р.

Керівник роботи канд. техн. наук, доцент Горбань Г. В.
(посада, прізвище, ім'я, по батькові) *(підпис)*
«24»січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок проходження онлайн-тестів»

Здобувач 409 гр,: Біба Євгеній Валерійович

Керівник: канд. техн. наук доцент Горбань Г. В.

Дана робота присвячена розробці програмного забезпечення, а саме платформи для створення онлайн-тестів.

У сучасному світі онлайн-освіта та дистанційне навчання набувають все більшої популярності, зокрема через глобальні зміни та потребу в адаптації до нових умов. Платформи для створення онлайн-тестів стають важливими інструментами для освітніх установ, корпоративного навчання та самоосвіти.

Об'єктом роботи є платформа для створення онлайн-тестів та її функціональні можливості.

Предметом роботи є процес проєктування та розробки платформи для створення онлайн-тестів, включаючи аналіз вимог до функціоналу та дизайну, реалізацію системи та її тестування.

Метою даної кваліфікаційної роботи є розробка та проєктування платформи для створення онлайн-тестів. Ця платформа надаватиме користувачам зручні інструменти для створення тестів, управління ними та аналізу результатів.

У першому розділі представлено аналіз предметної області, готових рішень та специфікацію вимог.

У другому розділі представлено розроблені діаграми варіантів використання, послідовності, станів.

У третьому розділі представлено модель та структуру бази даних системи, діаграму класів та мокап системи.

У четвертому розділі представлено реалізацію та функціонал платформи для створення онлайн-тестів.

КРБ викладена на 73 сторінки, вона містить 4 розділи, 38 ілюстрацій, 23 джерел в переліку посилань

Ключові слова: вебзастосунок, онлайн-тести, php, Laravel, CSS.

ABSTRACT

to the qualification work of the bachelor

"WebApp for passing online tests"

Student 409 gr,: Biba Yevhenii Valeriyovych

Supervisor: candidate technical Horban H.V., Associate Professor of Sciences

This work is devoted to the development of software, namely a platform for creating online tests.

In today's world, online education and distance learning are gaining more and more popularity, in particular due to global changes and the need to adapt to new conditions. Platforms for creating online tests are becoming important tools for educational institutions, corporate training and self-education.

The object of work is the platform for creating online tests and its functionality.

The subject of work is the process of designing and developing a platform for creating online tests, including analysis of functional and design requirements.

The purpose of work is to develop and design a platform for creating online tests. This platform will provide users with convenient tools for creating tests, managing them and analyzing the results.

The first section presents the analysis of the subject area, ready-made solutions and specification of requirements.

The second section presents the developed diagrams of use cases, sequences, and states.

The third section presents the system database model and structure, class diagram and system mockup.

The fourth chapter presents the implementation and functionality of the web application for passing online tests.

KRB is laid out on 73 pages, it contains 4 sections, 38 illustrations, 23 sources in the list of references

Keywords: web application, online tests , php, Laravel, CSS.

ЗМІСТ

ВСТУП

У сучасному світі онлайн-освіта стає все більш популярною, і однією з ключових її складових є ефективне тестування знань. Онлайн-тестування дозволяє знизити витрати на проведення екзаменів, підвищити об'єктивність оцінювання та забезпечити гнучкість у навчальному процесі.

В умовах глобальної пандемії COVID-19, потреба в таких платформах стала ще більш актуальною. Важливість розробки такої платформи також обумовлена зростаючими вимогами до інтерактивності, адаптивності та зручності користування, що сприяє покращенню якості освіти та розширенню доступу до навчальних ресурсів для широкого кола користувачів.

Об'єктом роботи є платформа для створення онлайн-тестів та її функціональні можливості.

Предметом роботи є процес проєктування та розробки платформи для створення онлайн-тестів, включаючи аналіз вимог до функціоналу та дизайну, реалізацію системи та її тестування.

Метою даної кваліфікаційної роботи є розробка та проєктування платформи для створення онлайн-тестів. Ця платформа надаватиме користувачам зручні інструменти для створення тестів, управління ними та аналізу результатів.

Завдання:

1. розробити функціональну та зручну платформу для створення онлайн-тестів;
2. спроєктувати базу даних для зберігання та обробки інформації про тести, запитання та результати;
3. оптимізувати швидкодію платформи та адаптувати її для різних пристроїв та екранів;
4. забезпечити можливість авторизації та реєстрації користувачів;
5. реалізувати функцію автоматичної перевірки відповідей та генерацію аналітичних звітів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

У сучасному світі освіти та бізнесу платформи для створення онлайн-тестів набувають все більшої популярності. Їх зростаюча популярність зумовлена прагненням до автоматизації процесів оцінювання знань, підвищення ефективності навчання та зменшення витрат часу на перевірку результатів. Розвиток цифрових технологій зробив такі платформи доступнішими і більш ефективними, що привернуло нових користувачів.

Ринок платформ для створення онлайн-тестів насичений різноманітними продуктами від великих міжнародних компаній до малих стартапів. Основними гравцями на ринку є такі компанії, як Google з Google Forms, Moodle, Kahoot! та Quizlet. Конкуренція в цій галузі є високою, і компанії постійно змагаються за увагу користувачів шляхом вдосконалення функціональності продуктів, розробки нових маркетингових стратегій та поліпшення взаємодії з клієнтами.

Цільова аудиторія платформ для створення онлайн-тестів є дуже різноманітною. Вона включає освітні заклади (школи, коледжі, університети), тренінгові компанії, HR-відділи підприємств, а також окремих викладачів та інструкторів. Освітні заклади використовують ці платформи для проведення іспитів, тестів і опитувань. Тренінгові компанії та HR-відділи застосовують їх для оцінювання знань і навичок співробітників, а також для сертифікації. Викладачі та інструктори використовують платформи для регулярного контролю знань учнів та студентів.

Платформи для створення онлайн-тестів надають широкий спектр функціональних можливостей. Це включає створення тестів з питаннями різних типів, таких як множинний вибір, правда/неправда, відкриті питання, завдання з короткою відповіддю тощо. Крім того, платформи дозволяють управляти тестами, зберігати їх у вигляді шаблонів для повторного використання, та надавати доступ до тестів через вебінтерфейс або мобільні застосунки. Важливою функцією є

автоматичне оцінювання результатів, генерація звітів і аналітичних даних, що дозволяє аналізувати успішність учасників та виявляти прогалини у знаннях.

Розвиток платформ для створення онлайн-тестів постійно вдосконалюється завдяки новим технологіям. Інновації включають використання штучного інтелекту для адаптивного навчання, інтеграцію з іншими освітніми системами (LMS), а також розвиток мобільних застосунків для зручності користувачів. Однією з ключових тенденцій є зростання популярності використання гейміфікації для підвищення залученості учасників. Крім того, спостерігається підвищення уваги до безпеки даних користувачів та забезпечення конфіденційності інформації.

Аналіз предметної області показує, що платформи для створення онлайн-тестів повинні поєднувати широкий функціонал, забезпечувати безпеку, бути масштабованими та зручними у використанні. Існуючі рішення мають свої переваги та недоліки, що створює можливості для розробки нових, більш досконалих платформ, які можуть запропонувати користувачам гнучкі та ефективні інструменти для тестування знань. Розробка нової платформи повинна враховувати всі ці аспекти, щоб задовольнити потреби широкого кола користувачів і бути конкурентоспроможною на ринку.

1.2 Аналіз готових рішень

На ринку існує багато платформ для створення онлайн-тестів, кожна з яких має свої переваги та недоліки.

Google Forms відзначається легкістю у використанні та інтеграцією з іншими продуктами Google, але має обмежені можливості для створення складних тестів та базову аналітику.

Moodle пропонує потужний набір функцій для управління навчальним процесом та інтеграцію з іншими освітніми інструментами, але є складним у налаштуванні та адмініструванні і вимагає хостингу.

Kahoot! приваблює своєю інтерактивністю та гейміфікацією, але обмежений у можливостях для серйозного академічного тестування, оскільки фокусується на

розважальних елементах. Quizlet забезпечує зручний інтерфейс для створення та обміну навчальними картками та має мобільний застосунок, проте обмежений функціонально для проведення тестів з великою кількістю учасників і не має розширених аналітичних функцій.

Розглянуто наступні вебсервіси:

- Google Forms;
- Moodle;
- Kahoot.

1.2.1 Аналіз Google Forms

Google Forms – це безкоштовний онлайн-інструмент від компанії Google, призначений для створення опитувань, анкет та тестів. Цей сервіс є частиною Google Drive і дозволяє користувачам створювати різноманітні форми для збору даних, які можуть бути зручним чином організовані та проаналізовані [1].

Головний екран даного готового рішення представлено на рис. 1.1.

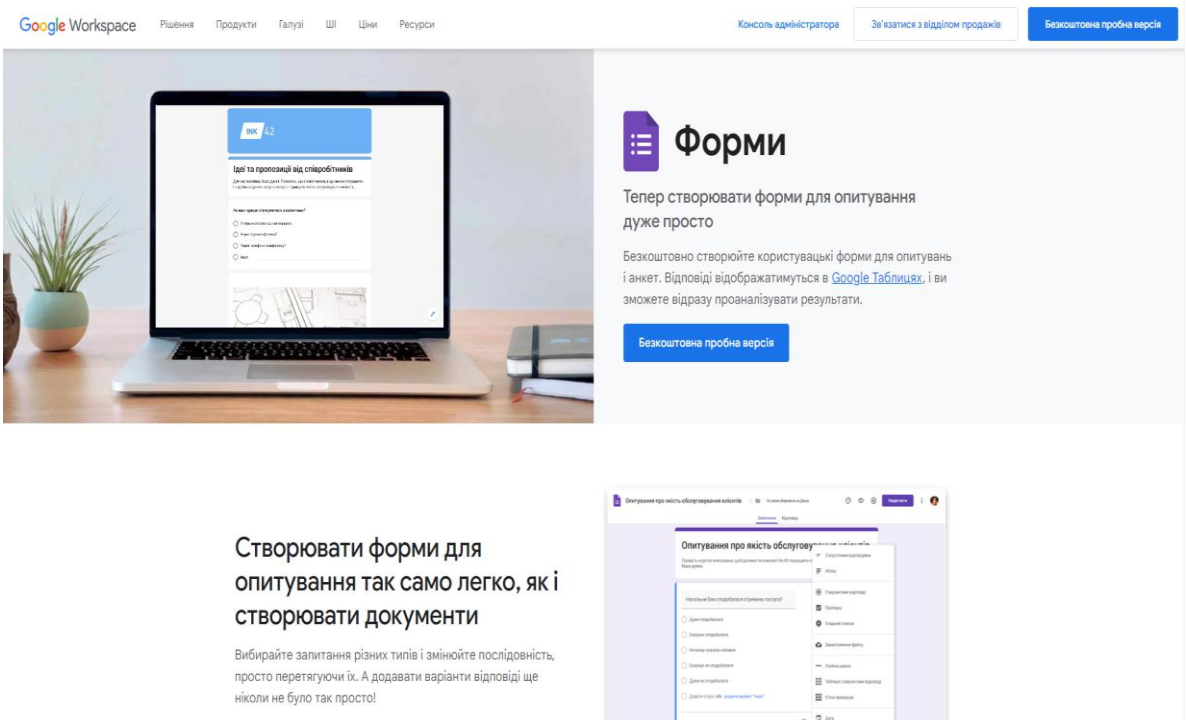


Рисунок 1.1 – Відображення головного екрану Google Forms

Основною перевагою Google Forms є його простота у використанні. Інтуїтивно зрозумілий інтерфейс дозволяє швидко створювати форми без

необхідності мати спеціальні знання чи навички в програмуванні. Крім того, інтеграція з іншими продуктами Google, такими як Google Sheets, дає можливість автоматично зберігати та аналізувати зібрані дані. Це робить Google Forms ідеальним вибором для швидкого та ефективного збору інформації.

Ще однією вагомою перевагою є безкоштовність сервісу. Google Forms доступний для всіх користувачів Google без додаткових витрат, що робить його особливо привабливим для невеликих організацій, навчальних закладів та індивідуальних користувачів.

Однак, Google Forms має й певні недоліки. Основним обмеженням є його базовий функціонал, який не завжди відповідає потребам користувачів, що вимагають складних рішень для проведення тестування. Наприклад, можливості налаштування дизайну форм та створення складних логічних умов є досить обмеженими. Це може стати проблемою для тих, хто шукає більш просунуті функції та гнучкість у створенні тестів.

Також слід зазначити, що аналітичні інструменти Google Forms є досить простими. Для глибшого аналізу зібраних даних користувачам часто доводиться експортувати дані до інших програм, таких як Microsoft Excel чи спеціалізовані системи аналізу даних. Це додає додаткові кроки у процесі обробки інформації.

Таким чином, Google Forms є потужним інструментом для створення опитувань та тестів, що поєднує простоту використання та інтеграцію з іншими продуктами Google. Проте його базовий функціонал і обмежені аналітичні можливості можуть бути недостатніми для деяких користувачів, які потребують більш гнучких та потужних рішень.

1.2.2 Аналіз Moodle

Moodle – це безкоштовна платформа для управління навчанням (LMS), яка використовується для створення онлайн-курсів, проведення дистанційного навчання та управління освітніми процесами. Moodle є відкритим програмним забезпеченням, що означає, що його код можна вільно використовувати, змінювати та розповсюджувати [2].

Головний екран даного готового рішення представлено на рис. 1.2.

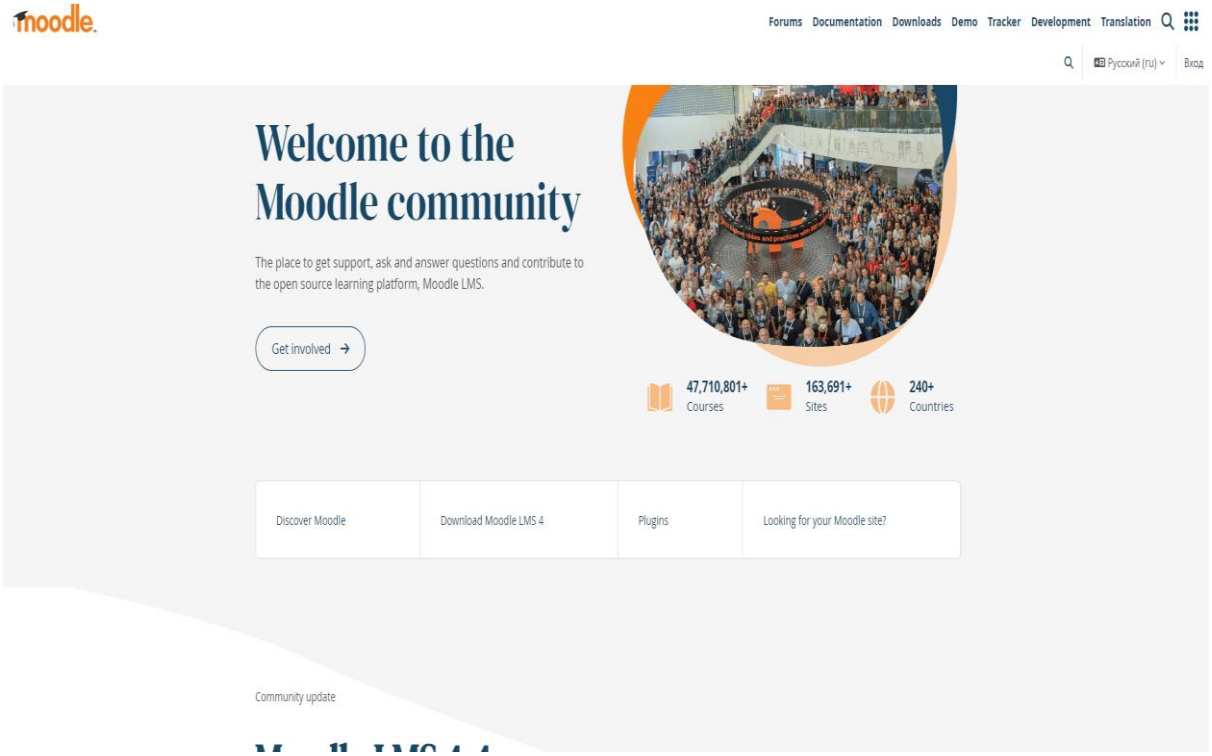


Рисунок 1.2 – Відображення головного екрану Moodle

Однією з основних переваг Moodle є його потужний набір функцій, який дозволяє створювати і керувати навчальними матеріалами, тестами, форумами, завданнями та багатьма іншими видами діяльності. Це робить його універсальним інструментом, який підходить для різних типів навчальних закладів, від шкіл до університетів. Завдяки відкритому коду, Moodle може бути налаштований та розширений відповідно до специфічних потреб користувачів, що забезпечує високий рівень гнучкості та адаптивності.

Ще однією важливою перевагою Moodle є його активна спільнота користувачів і розробників. Вони постійно працюють над покращенням платформи, додаючи нові функції, виправляючи помилки та розробляючи плагіни, які розширюють можливості системи. Це забезпечує постійний розвиток і оновлення платформи, роблячи її більш надійною та функціональною.

Проте Moodle має і певні недоліки. Основним викликом для користувачів може бути його складність у налаштуванні та адмініструванні. Для повноцінного використання всіх можливостей системи потрібні знання в області IT та досвід

адміністрування серверів. Це може створювати труднощі для невеликих організацій або окремих викладачів, які не мають достатньо ресурсів для технічної підтримки.

Ще одним недоліком є потреба у хостингу. Moodle потребує серверного середовища для роботи, що означає додаткові витрати на хостинг та технічне обслуговування. Крім того, попри широкий функціонал, інтерфейс платформи може здаватися застарілим або незручним для деяких користувачів, що може вплинути на їх досвід використання системи.

Таким чином, Moodle є потужним та гнучким інструментом для управління навчальними процесами, який пропонує широкий набір функцій і можливостей для налаштування. Однак складність налаштування, потреба у технічній підтримці та вимоги до хостингу можуть бути значними перепонами для деяких користувачів.

1.2.3 Аналіз Kahoot

Kahoot! – це платформа для створення інтерактивних тестів, опитувань та навчальних ігор, яка дозволяє проводити навчальні та розважальні заходи в режимі реального часу. Використовуючи принцип гейміфікації, Kahoot! робить навчання більш цікавим та залучаючим для учасників. Ця платформа широко використовується в школах, університетах та на підприємствах для навчання та перевірки знань [3].

Головний екран даного готового рішення представлено на рис. 1.3.

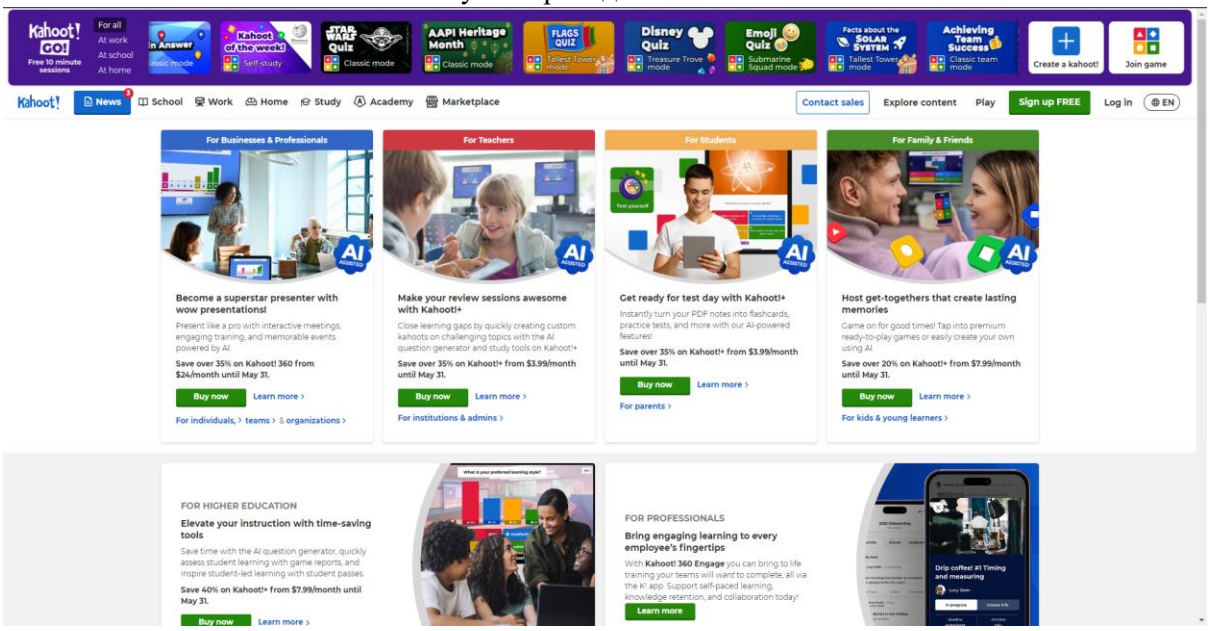


Рисунок 1.3 – Відображення головного екрану Kahoot

Однією з головних переваг Kahoot! є його здатність перетворювати навчання на захопливу гру. Інтерактивність платформи стимулює активну участь учнів або співробітників, що підвищує їхню залученість і мотивацію. Простий та зрозумілий інтерфейс дозволяє швидко створювати тести та опитування, навіть без спеціальних знань чи навичок. Крім того, платформа підтримує мобільні пристрої, що робить її зручною для використання у будь-якому місці і в будь-який час.

Kahoot! також забезпечує можливість миттєвого зворотного зв'язку, що дозволяє учасникам відразу дізнаватися результати своїх відповідей і таким чином покращувати свої знання в режимі реального часу. Викладачі та організатори можуть легко відстежувати прогрес учасників за допомогою звітів і аналітичних інструментів, що допомагає виявляти слабкі місця у знаннях та адаптувати навчальний процес.

Проте Kahoot! має свої недоліки. Одним із основних обмежень є його фокус на гейміфікації, що може здаватися занадто розважальним для деяких типів навчання або професійного тестування. Інтерфейс, орієнтований на гру, може не відповідати серйозним академічним або корпоративним потребам, де потрібен більш формальний підхід. Крім того, хоча базові функції платформи безкоштовні,

деякі розширені можливості та функції доступні лише в платній версії, що може обмежити доступ до повного спектра інструментів.

Таким чином, Kahoot! є ефективним інструментом для створення інтерактивних ігрових навчальних заходів, які підвищують залученість і мотивацію учасників. Однак його гейміфікаційний підхід може бути не завжди доречним для серйозних академічних або професійних контекстів, і деякі функції можуть вимагати платної підписки.

1.3 Специфікація вимог до програмного забезпечення

Призначення системи (застосунку), для якої розробляється програмне забезпечення:

Розробка програмного забезпечення для створення функціональної та зручної платформи, спеціалізованої на створенні онлайн-тестів.

Погодження, що ухвалені в програмній документації:

Наразі погодження в програмній документації відсутні, що може вплинути на наступні кроки у розробці.

Межі проєкту ПЗ:

Сфера застосування включає весь цикл життя інтернет-платформи, від створення до управління та підтримки.

Сфера застосування:

Розробка платформи для створення онлайн-тестів.

Характеристики користувачів:

Різноманітність користувачів, від звичайних користувачів (вчителі, студенти чи учні) до адміністраторів платформи.

Загальна структура і склад системи:

Включає функціонал платформи, управлінські інструменти, систему створення тестів, систему аналітики тестів, базу даних.

Загальні обмеження:

Технічні та правові обмеження, пов'язані з реалізацією інтернет-платформи.

Функції вебзастосунку:

Реєстрація та авторизація користувачів, списку тестів, створення тестів, тощо.

Джерела і зміст вхідної інформації (даних):

Інформація про тести, аналітику та клієнтів.

Мова і технологія розробки ПЗ:

Використання технологій Php, Laravel, mysql, CSS, JS, HTML для реалізації проєкту.

Інтерфейс користувача:

Розробка зручного та інтуїтивно зрозумілого інтерфейсу для користувачів платформи.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра розглянуто платформу для створення онлайн-тестів, який пропонує зручний підхід до створення різного роду тестів.

Описано функціонал платформи, що спрямований на полегшення ведення навчального процесу вчителям.

В аналізі існуючих систем також виявлено ключові аспекти та вдалі рішення, які можна врахувати під час розробки власної платформи. Проаналізовані Google Forms, Moodle та Kahoot.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ОНЛАЙН-ТЕСТІВ

2.1 Розробка діаграми варіантів використання

Діаграма варіантів використання (Use Case Diagram) - це графічне зображення функціональних вимог до системи, яке показує, які дії можуть виконувати актори (користувачі або системи) у взаємодії з системою. Основна мета діаграми варіантів використання - це визначення функціональних можливостей системи та її взаємодії з користувачами.

Діаграма варіантів використання платформи, представлено на рис. 2.1.

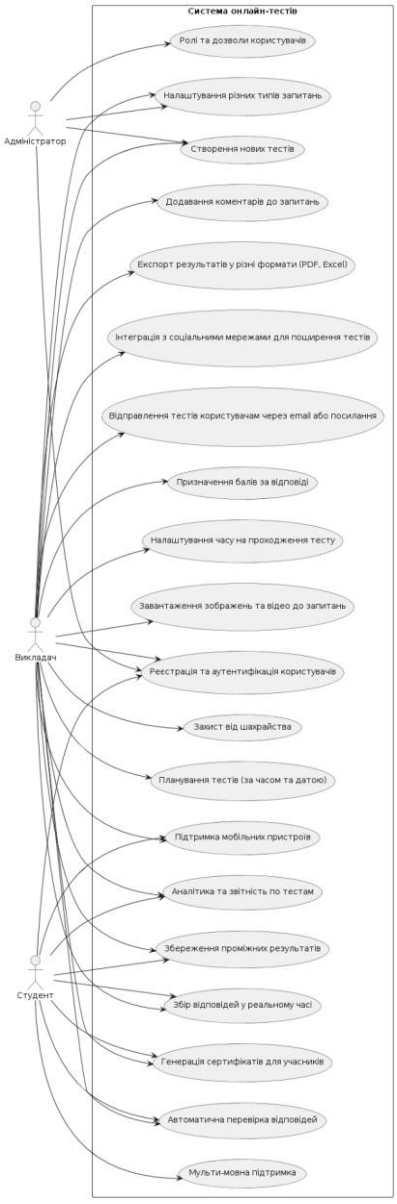


Рисунок 2.1 – Діаграма варіантів використання вебсайту

Дана діаграма варіантів використання відображає усі сценарії роботи програми

2.2 Розробка діаграми класів

Діаграма класів є одним з основних видів діаграм у мові моделювання UML (Unified Modeling Language). Вона використовується для візуалізації структури програмного коду, зокрема класів, їх атрибутів, методів та взаємозв'язків між класами.

Діаграму класів, представлено на рис. 2.2.

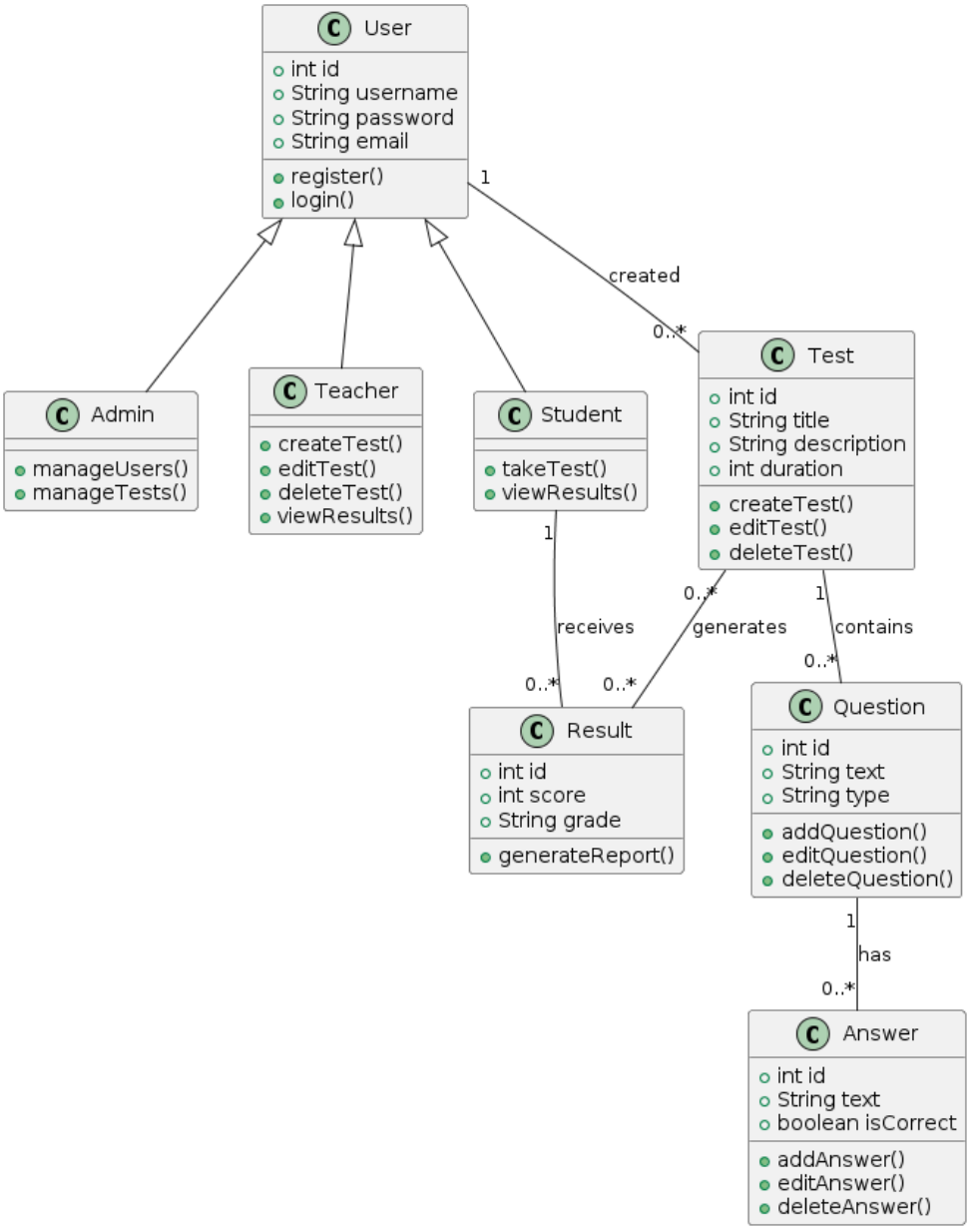


Рисунок 2.2 – Діаграма класів вебсайту

Ця діаграма описує такі класи:

- User (базовий клас для користувачів);
- Підкласи: Admin, Teacher, Student;
- Test (тести, створювані викладачами);
- Question (запитання, що входять до тесту);
- Answer (відповіді на запитання);
- Result (результати тестування).

Зв'язки між класами:

- Один користувач може створювати багато тестів;
- Один тест може містити багато запитань;
- Одне запитання може мати багато відповідей;
- Тест може генерувати багато результатів;
- Один студент може мати багато результатів.

2.3 Розробка діаграми послідовностей дій системи

Діаграма послідовності (Sequence Diagram) — це вид діаграми UML (Unified Modeling Language), яка використовується для відображення послідовності обміну повідомленнями між об'єктами або компонентами системи в певному сценарії. Вона показує, як і в якому порядку взаємодіють між собою різні частини системи для виконання певного процесу або функції

Діаграма послідовностей системи представлена на рис. 2.3.

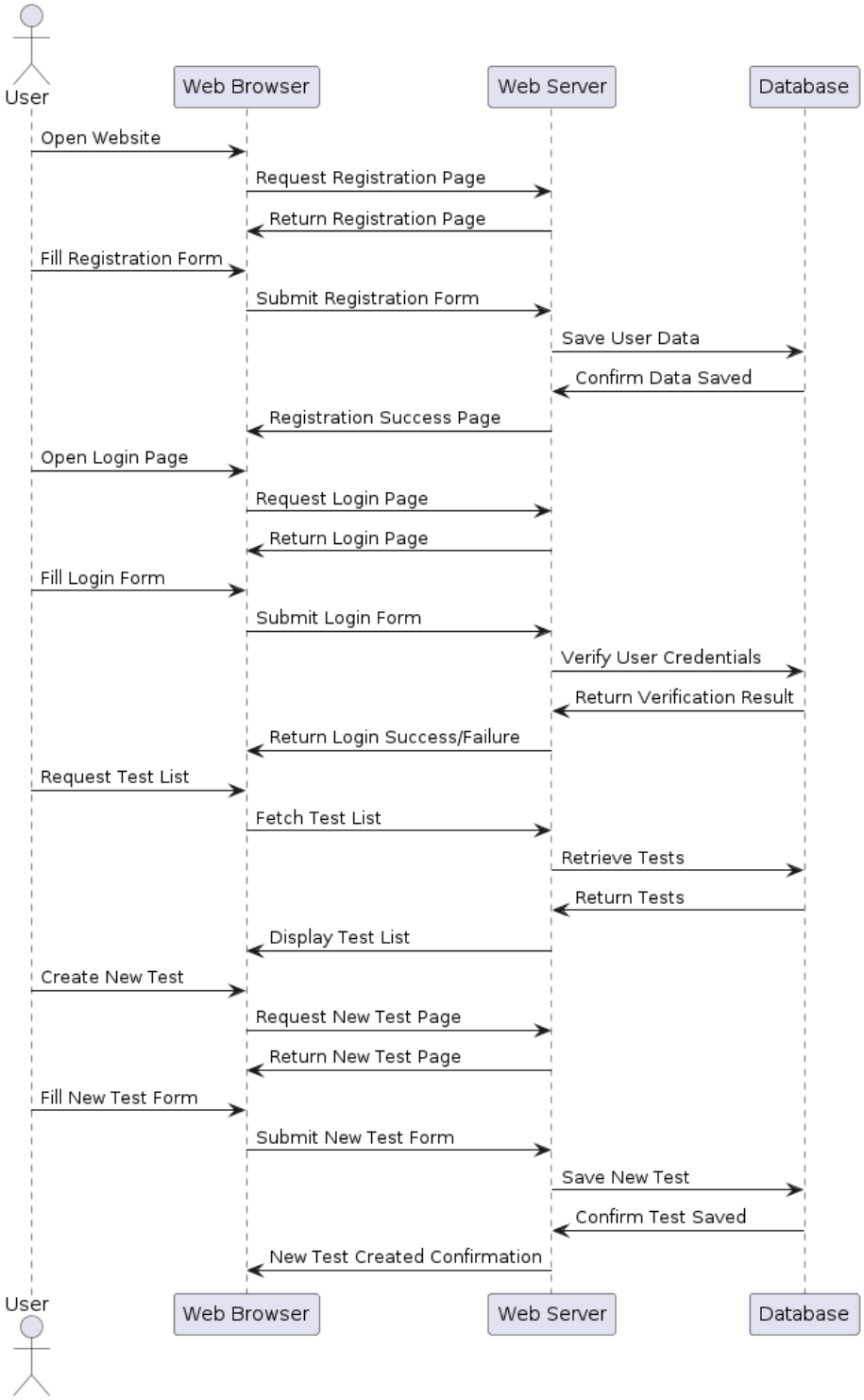


Рисунок 2.3 – Діаграма послідовностей

Загалом, діаграма послідовностей допомогла виявити і документувати основні сценарії використання системи, підкреслити важливі аспекти взаємодії компонентів і забезпечити основу для подальшого аналізу та покращення системи.

2.4 Розробка діаграми розгортання

Діаграма розгортання (deployment diagram) показує фізичну архітектуру системи, тобто, де і як розміщені різні компоненти системи в реальному середовищі.

Діаграма розгортання системи представлена на рис. 2.4.

Рисунок 2.4 – Діаграма розгортання системи

Діаграма вказує на архітектуру системи, вебзастосунок працює на наявність клієнтської частини, серверної частини та бази даних.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра описано проєктування платформи для створення онлайн-тестів. Під цією метою були розроблені та детально розглянуті наступні моделі та діаграми:

- діаграма варіантів використання системи;
- діаграма класів системи;
- діаграма розгортання.

Ці моделі та діаграми були створені для забезпечення чіткого розуміння архітектури та функціональності розробленого платформи для створення онлайн-тестів. Вони є ключовими елементами для подальшого розвитку та реалізації проєкту.

3 ОПИС ТЕХНОЛОГІЙ РОЗРОБКИ ТА МОДЕЛЮВАННЯ БАЗИ ДАНИХ

3.1 Опис технологій розробки

MySQL – це одна з найпопулярніших систем управління реляційними базами даних (СУРБД) у світі, яка широко використовується для різних видів застосунків, починаючи від невеликих вебсайтів до великих корпоративних систем. MySQL є частиною відкритого програмного забезпечення і пропонує високий рівень продуктивності, надійності та масштабованості, що робить її привабливою для багатьох розробників [7,18].

Один з головних аспектів, що робить MySQL привабливою, – це її висока продуктивність. MySQL здатна обробляти великі обсяги даних з мінімальними затримками, що є критично важливим для застосунків, які вимагають швидкого доступу до даних. Висока продуктивність досягається за рахунок оптимізації запитів та ефективного управління пам'яттю. MySQL підтримує індексацію, що значно прискорює пошук і доступ до даних. Вона також має вбудовані механізми кешування, які допомагають зменшити навантаження на систему та підвищити швидкість обробки запитів.

MySQL є надзвичайно надійною СУРБД, що забезпечується через її архітектуру та інструменти для резервного копіювання та відновлення. Вона підтримує транзакції, що дозволяє забезпечити цілісність даних у разі збоїв або одночасного доступу з боку декількох користувачів. Крім того, MySQL має функції відновлення після збоїв, що дозволяє швидко відновити роботу системи у разі непередбачених ситуацій.

Однією з ключових переваг MySQL є її масштабованість. Вона може ефективно працювати як на одному сервері, так і в кластері, що дозволяє забезпечити безперебійну роботу застосунка при зростанні кількості користувачів і обсягу даних. MySQL підтримує реплікацію, яка дозволяє розподіляти навантаження між декількома серверами та забезпечувати високу доступність даних.

MySQL є відкритим програмним забезпеченням з відкритим кодом, що дає розробникам можливість вносити зміни в код та адаптувати систему під свої потреби. Це дозволяє значно знизити витрати на розробку та підтримку програмного забезпечення. Відкрита ліцензія також дозволяє розробникам вільно використовувати MySQL без необхідності сплачувати за ліцензії, що робить її доступною для малих і середніх підприємств.

Серед інших важливих особливостей MySQL слід зазначити її гнучку систему авторизації користувачів та контролю доступу, що дозволяє забезпечити високий рівень безпеки даних. MySQL підтримує різні рівні привілеїв для користувачів, що дозволяє точно контролювати доступ до даних та операцій. Крім того, MySQL має вбудовані засоби для захисту від SQL-ін'єкцій, що є однією з найпоширеніших загроз для вебзастосунків.

MySQL має широку підтримку спільноти розробників, що дозволяє отримувати швидку допомогу та консультації. Величезна кількість документації, форумів, навчальних матеріалів та прикладів коду дозволяє швидко розібратися в особливостях роботи з MySQL та знаходити рішення для різних задач. Крім того, MySQL регулярно оновлюється та вдосконалюється, що забезпечує її відповідність сучасним вимогам та стандартам.

Одним з прикладів успішного використання MySQL є її інтеграція з іншими популярними технологіями, такими як PHP та Apache, що дозволяє створювати потужні вебзастосунки. В поєднанні з PHP, MySQL забезпечує швидкий і зручний доступ до бази даних, що дозволяє розробникам створювати динамічні вебсайти з високою продуктивністю. Apache, як вебсервер, забезпечує стабільну роботу застосунка та обробку великої кількості запитів одночасно.

Загалом, вибір MySQL як основної СУРБД для проєкту обумовлений її численними перевагами, такими як висока продуктивність, надійність, масштабованість, безпека та доступність. MySQL є потужним інструментом для управління даними, який дозволяє створювати ефективні та надійні вебзастосунки, що відповідають сучасним вимогам та стандартам. Вибір MySQL

також підкріплюється її широкою підтримкою спільноти розробників та відкритою ліцензією, що робить її доступною для широкого кола користувачів.

PHP – це один із найпопулярніших мов програмування для веброзробки, яка використовується для створення динамічних вебсайтів та вебзастосунків. PHP, або Hypertext Preprocessor, була створена у 1994 році і з того часу здобула величезну популярність завдяки своїй простоті, гнучкості та широкому спектру можливостей [4-9, 20-21].

Однією з головних причин вибору PHP є її простота та доступність. Мова має низький поріг входу, що дозволяє швидко навчитися основам і почати розробку. Синтаксис PHP дуже схожий на C та Perl, що робить її легкою для освоєння для програмістів, знайомих з цими мовами. Крім того, PHP добре задокументована, і для неї існує безліч навчальних ресурсів, що спрощує процес навчання та пошуку рішень під час розробки.

PHP має високу сумісність із різними базами даних, такими як MySQL, PostgreSQL, SQLite та інші. Це дозволяє розробникам вибирати найбільш підходящу СУРБД для свого проєкту і легко інтегрувати її з PHP-застосунком. Крім того, PHP підтримує широкий спектр вебсерверів, таких як Apache, Nginx та IIS, що забезпечує гнучкість у виборі платформи для розгортання застосунка.

Мова PHP має високу продуктивність і може ефективно обробляти великі обсяги запитів одночасно. Це досягається завдяки вбудованим механізмам кешування та оптимізації коду. Крім того, PHP підтримує багатопоточність, що дозволяє зменшити час виконання складних операцій і підвищити швидкість роботи застосунка. PHP також підтримує роботу з різними форматами даних, такими як XML, JSON, що робить її універсальною для розробки вебзастосунків.

Ще однією важливою перевагою PHP є її підтримка великої кількості фреймворків і бібліотек, що значно спрощує процес розробки та дозволяє використовувати готові рішення для реалізації різних функціональних можливостей. Наприклад, такі фреймворки, як Laravel, Symfony, CodeIgniter, надають розробникам потужні інструменти для створення складних і масштабованих вебзастосунків. Ці фреймворки включають в себе різні модулі та

бібліотеки, які допомагають скоротити час розробки та забезпечити високу якість коду.

PHP також підтримує інтеграцію з іншими мовами програмування та технологіями, такими як JavaScript, HTML, CSS, що дозволяє створювати інтерактивні та динамічні інтерфейси користувача. Це особливо важливо в умовах сучасних вебзастосунків, де користувачі очікують високого рівня взаємодії та зручності використання. Застосування PHP у поєднанні з JavaScript-фреймворками, такими як React або Angular, дозволяє створювати потужні односторінкові застосунки (SPA) з високою продуктивністю та швидким часом завантаження.

PHP має вбудовані засоби для забезпечення безпеки вебзастосунків. Вона включає в себе функції для захисту від SQL-ін'єкцій, міжсайтових скриптових атак (XSS) і підробки міжсайтових запитів (CSRF). Це дозволяє розробникам зосередитися на функціональності застосунка, не турбуючись про можливі вразливості. Крім того, PHP підтримує шифрування даних та роботу з SSL-сертифікатами, що забезпечує безпеку передавання даних між сервером і клієнтом.

PHP має широку підтримку спільноти розробників, що дозволяє отримувати швидку допомогу та консультації. Величезна кількість документації, форумів, навчальних матеріалів та прикладів коду дозволяє швидко розібратися в особливостях роботи з PHP та знаходити рішення для різних задач. Крім того, PHP регулярно оновлюється та вдосконалюється, що забезпечує її відповідність сучасним вимогам та стандартам.

Одним з прикладів успішного використання PHP є її інтеграція з іншими популярними технологіями, такими як MySQL та Apache, що дозволяє створювати потужні вебзастосунки. В поєднанні з MySQL, PHP забезпечує швидкий і зручний доступ до бази даних, що дозволяє розробникам створювати динамічні вебсайти з високою продуктивністю. Apache, як вебсервер, забезпечує стабільну роботу застосунка та обробку великої кількості запитів одночасно.

Загалом, вибір PHP як основної мови програмування для веброзробки обумовлений її численними перевагами, такими як простота, гнучкість, висока продуктивність, безпека та підтримка широкого спектру інструментів і фреймворків. PHP є потужним інструментом для створення динамічних і інтерактивних вебзастосунків, що відповідають сучасним вимогам та стандартам. Вибір PHP також підкріплюється її широкою підтримкою спільноти розробників та відкритою ліцензією, що робить її доступною для широкого кола користувачів.

Laravel – це один із найпопулярніших і потужних PHP-фреймворків для веброзробки, який був створений для того, щоб зробити процес розробки зручнішим, швидшим і продуктивнішим. Laravel пропонує численні переваги, які роблять його вибір обґрунтованим для багатьох проєктів, включаючи високу продуктивність, гнучкість, зручність використання та широкий спектр інструментів [10-17, 22].

Однією з ключових особливостей Laravel є його чітка і зрозуміла архітектура, яка базується на принципі Model-View-Controller (MVC). Це дозволяє розділити логіку програми на окремі компоненти, що робить код більш структурованим, зрозумілим і легким у підтримці. MVC-архітектура також сприяє повторному використанню коду, що знижує час і витрати на розробку. Крім того, така архітектура забезпечує чітке розділення відповідальностей, що спрощує тестування і відлагодження коду.

Laravel надає розробникам широкий спектр інструментів і функціональних можливостей, які спрощують процес створення вебзастосунків. Однією з таких можливостей є система маршрутизації, яка дозволяє легко управляти URL-запитами і визначати, які контролери і методи будуть обробляти певні запити. Це дозволяє створювати гнучкі та масштабовані вебзастосунки, які легко адаптуються до змін у вимогах користувачів і бізнесу.

Ще однією важливою особливістю Laravel є його підтримка об'єктно-реляційного відображення (ORM) через Eloquent ORM. Це дозволяє працювати з базою даних за допомогою об'єктно-орієнтованого підходу, що значно спрощує процес написання запитів і маніпуляції даними. Eloquent ORM підтримує зв'язки

між таблицями, що дозволяє легко працювати зі складними структурами даних. Крім того, Eloquent надає зручні методи для роботи з базою даних, що робить код більш читабельним і зрозумілим.

Laravel також надає потужний інструмент для міграцій баз даних, який дозволяє легко керувати змінами в структурі бази даних. Це особливо корисно при роботі в команді, де різні розробники можуть вносити зміни в базу даних. Міграції дозволяють синхронізувати зміни і забезпечують цілісність даних. Крім того, Laravel має вбудовані засоби для забезпечення безпеки вебзастосунків, такі як захист від SQL-ін'єкцій, міжсайтових скриптових атак (XSS) і підробки міжсайтових запитів (CSRF). Це дозволяє розробникам зосередитися на функціональності застосунка, не турбуючись про можливі вразливості.

Laravel має потужну систему аутентифікації та авторизації, що дозволяє легко додавати функціональність для реєстрації та входу користувачів, а також керувати їхніми правами доступу. Це особливо важливо для застосунків, які вимагають високого рівня безпеки та захисту даних. Крім того, Laravel підтримує роботу з API, що дозволяє створювати застосунки, які можуть взаємодіяти з іншими системами та сервісами.

Однією з найпотужніших функцій Laravel є підтримка шаблонів іменованих завдань (Jobs) та черг (Queues), що дозволяє виконувати фонові завдання і обробляти великі обсяги даних без зупинки основного потоку застосунка. Це забезпечує високу продуктивність та ефективність роботи застосунка, особливо при високих навантаженнях.

Laravel також має широкий спектр додаткових пакетів і модулів, які розширюють його функціональність. Наприклад, Laravel Horizon дозволяє моніторити черги завдань, Laravel Passport забезпечує аутентифікацію через OAuth2, а Laravel Scout дозволяє додавати функціональність пошуку. Це дозволяє розробникам легко інтегрувати додаткові можливості в свої застосунки без необхідності писати власні рішення з нуля.

Загалом, вибір Laravel як фреймворку для веброзробки обумовлений його численними перевагами, такими як висока продуктивність, гнучкість, зручність

використання та широкий спектр інструментів і функціональних можливостей. Laravel є потужним інструментом для створення складних і масштабованих вебзастосунків, що відповідають сучасним вимогам та стандартам. Вибір Laravel також підкріплюється його широкою підтримкою спільноти розробників та регулярними оновленнями, що забезпечує його відповідність сучасним вимогам та стандартам.

OpenServer – це програмний комплекс, який включає всі необхідні інструменти для розробки, тестування і розгортання вебзастосунків на локальному комп'ютері. Він є одним з найпопулярніших рішень серед розробників завдяки своїй простоті використання, гнучкості налаштувань та підтримці великої кількості програмних компонентів.

Однією з головних переваг OpenServer є його простота установки і налаштування. Програмний комплекс поставляється у вигляді готового до використання пакету, який не вимагає додаткових налаштувань. Це дозволяє швидко розпочати роботу і зосередитися на розробці застосунка, а не на конфігурації середовища. Встановлення OpenServer займає лише кілька хвилин і не потребує глибоких знань у налаштуванні серверного програмного забезпечення.

OpenServer включає в себе такі компоненти, як вебсервер Apache або Nginx, СУРБД MySQL або PostgreSQL, а також інтерпретатор PHP. Це дозволяє створити повноцінне середовище розробки, яке максимально наближене до бойового сервера. Крім того, OpenServer підтримує різні версії PHP і баз даних, що дозволяє тестувати застосунки в різних умовах і забезпечувати їх сумісність. Це особливо важливо для проєктів, які мають працювати на різних серверних конфігураціях.

OpenServer має зручний графічний інтерфейс, який дозволяє легко керувати сервісами, налаштовувати конфігурації та відслідковувати стан системи. Це значно спрощує процес адміністрування і дозволяє швидко вносити зміни в налаштування середовища розробки. Крім того, OpenServer підтримує автоматичний запуск сервісів при завантаженні операційної системи, що

забезпечує швидкий старт роботи без необхідності вручну запускати кожен компонент.

Ще однією важливою перевагою OpenServer є його підтримка різних інструментів для розробки і тестування вебзастосунків. Вона включає в себе вбудовані редактори коду, інструменти для роботи з базами даних, а також засоби для відлагодження і профілювання коду. Це дозволяє розробникам ефективно працювати з кодом, швидко знаходити і виправляти помилки, а також оптимізувати продуктивність застосунка. Крім того, OpenServer підтримує інтеграцію з різними системами контролю версій, такими як Git, що дозволяє ефективно управляти змінами в коді і працювати в команді.

Однією з ключових особливостей OpenServer є його гнучкість у налаштуваннях. Користувачі можуть легко додавати або видаляти компоненти, змінювати конфігурації сервісів та налаштовувати середовище розробки під свої потреби. Це дозволяє створити оптимальне середовище для розробки, яке максимально відповідає вимогам конкретного проєкту. Крім того, OpenServer підтримує створення віртуальних хостів, що дозволяє розробляти і тестувати декілька проєктів одночасно на одному комп'ютері.

OpenServer також має вбудовані інструменти для моніторингу і аналізу продуктивності застосунків. Це дозволяє виявляти і усувати вузькі місця у коді, оптимізувати роботу застосунка і забезпечувати його стабільну роботу при високих навантаженнях. Крім того, OpenServer підтримує автоматичне оновлення компонентів, що дозволяє завжди мати актуальну версію програмного забезпечення і використовувати останні можливості та виправлення безпеки.

Важливою перевагою OpenServer є його сумісність з різними операційними системами, включаючи Windows та Linux. Це дозволяє використовувати OpenServer на різних платформах і забезпечувати однакові умови для розробки і тестування застосунків. Крім того, OpenServer підтримує роботу в режимі командного рядка, що дозволяє автоматизувати процеси розгортання і налаштування середовища.

OpenServer має широку підтримку спільноти розробників, що дозволяє отримувати швидку допомогу та консультації. Величезна кількість документації, форумів, навчальних матеріалів та прикладів коду дозволяє швидко розібратися в особливостях роботи з OpenServer та знаходити рішення для різних задач. Крім того, OpenServer регулярно оновлюється та вдосконалюється, що забезпечує його відповідність сучасним вимогам та стандартам.

Загалом, вибір OpenServer як середовища розробки обумовлений його численними перевагами, які включають простоту використання, гнучкість налаштувань, підтримку великої кількості програмних компонентів і інструментів для моніторингу та аналізу продуктивності. OpenServer є потужним інструментом для розробки і тестування вебзастосунків, який дозволяє створювати ефективні і надійні рішення, що відповідають сучасним вимогам та стандартам. Вибір OpenServer також підкріплюється його широкою підтримкою спільноти розробників та регулярними оновленнями, що забезпечує його відповідність сучасним вимогам та стандартам.

3.2 Моделювання бази даних

Фізична модель бази даних (БД) є конкретним представленням даних у базі даних, включаючи структуру таблиць, індексів, обмежень та інших елементів, що зберігаються у фізичній пам'яті комп'ютера. Це рівень моделювання, на якому реалізуються логічні структури даних у вигляді фізичних файлів на дисках, що робить їх доступними для обробки запитів і виконання операцій у реальному часі. У даній статті буде розглянуто детально всі аспекти фізичної моделі БД, її значення, компоненти та підходи до проєктування [23].

Фізична модель бази даних сайту для створення тестів представлено на рис. 3.1.

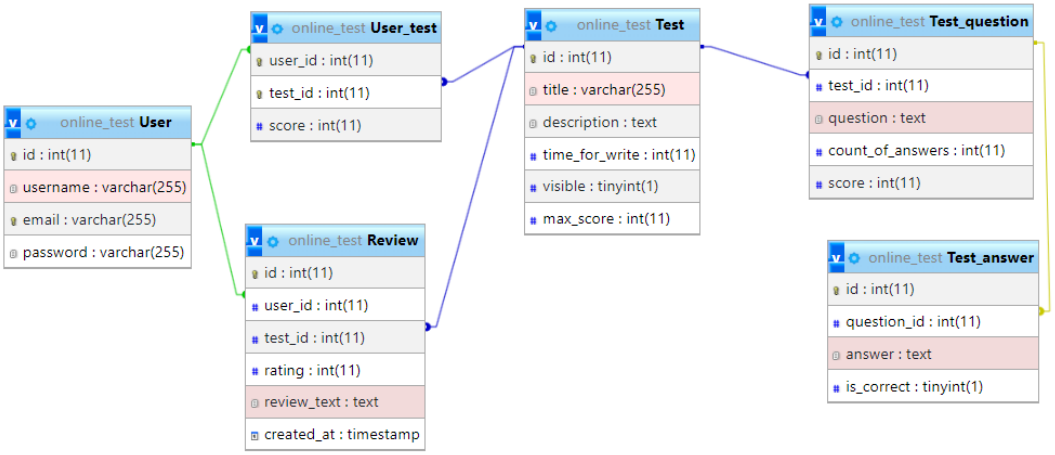


Рисунок 3.1 – Фізична модель бази даних

Розглянемо кожну таблицю окремо.

Таблиця User

Призначення:

Таблиця User відповідає за зберігання інформації про користувачів, які реєструються та користуються системою для створення та проходження тестів.

Поля:

– id (INT, AUTO_INCREMENT, PRIMARY KEY): Унікальний ідентифікатор користувача.

– username (VARCHAR(255), NOT NULL): Ім'я користувача.

– email (VARCHAR(255), NOT NULL, UNIQUE): Електронна пошта користувача. Вона повинна бути унікальною.

– password (VARCHAR(255), NOT NULL): Пароль користувача, зазвичай зберігається в захищеному вигляді (хешованому).

Структуру таблиці представлено на рис. 3.2.

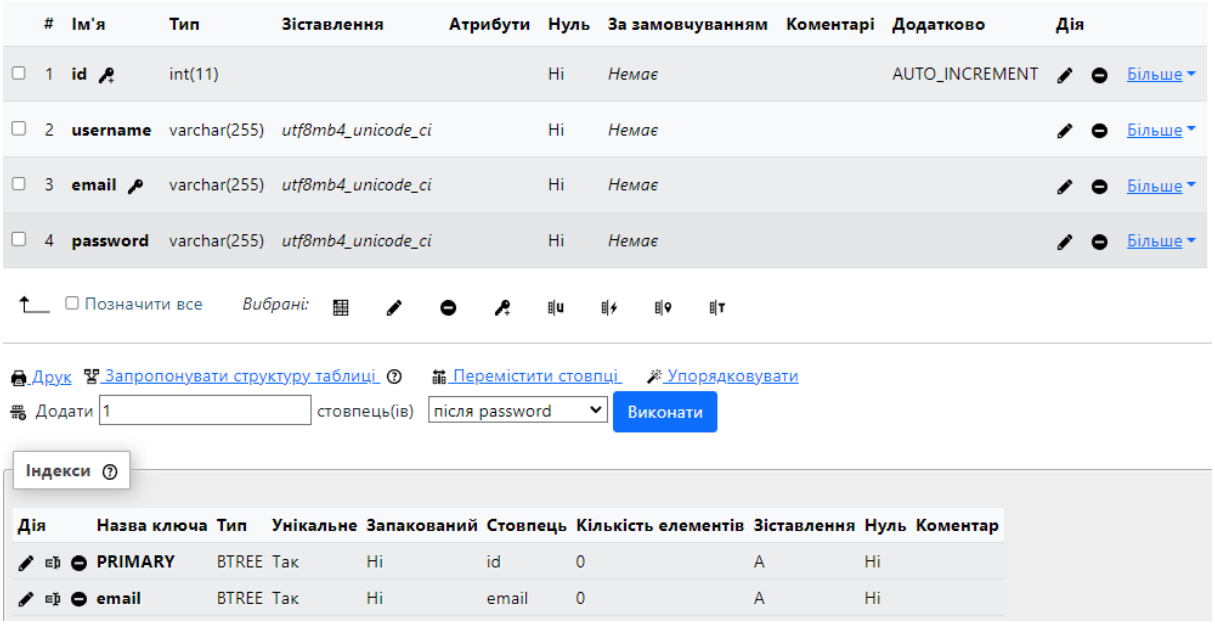


Рисунок 3.2 – Структура таблиці користувачів

Таблиця Test

Призначення:

Таблиця Test відповідає за зберігання інформації про створені тести.

Поля:

– id (INT, AUTO_INCREMENT, PRIMARY KEY): Унікальний ідентифікатор тесту.

– title (VARCHAR(255), NOT NULL): Назва тесту.

– description (TEXT): Опис тесту, що може включати додаткову інформацію для учасників.

– time_for_write (INT, NOT NULL): Час, відведений на проходження тесту (в хвилинах).

– visible (BOOLEAN, DEFAULT TRUE): Вказує, чи є тест видимим для користувачів.

– max_score (INT, NOT NULL): Максимальна кількість балів, яку можна отримати за проходження тесту.

Структуру таблиці представлено на рис. 3.3.

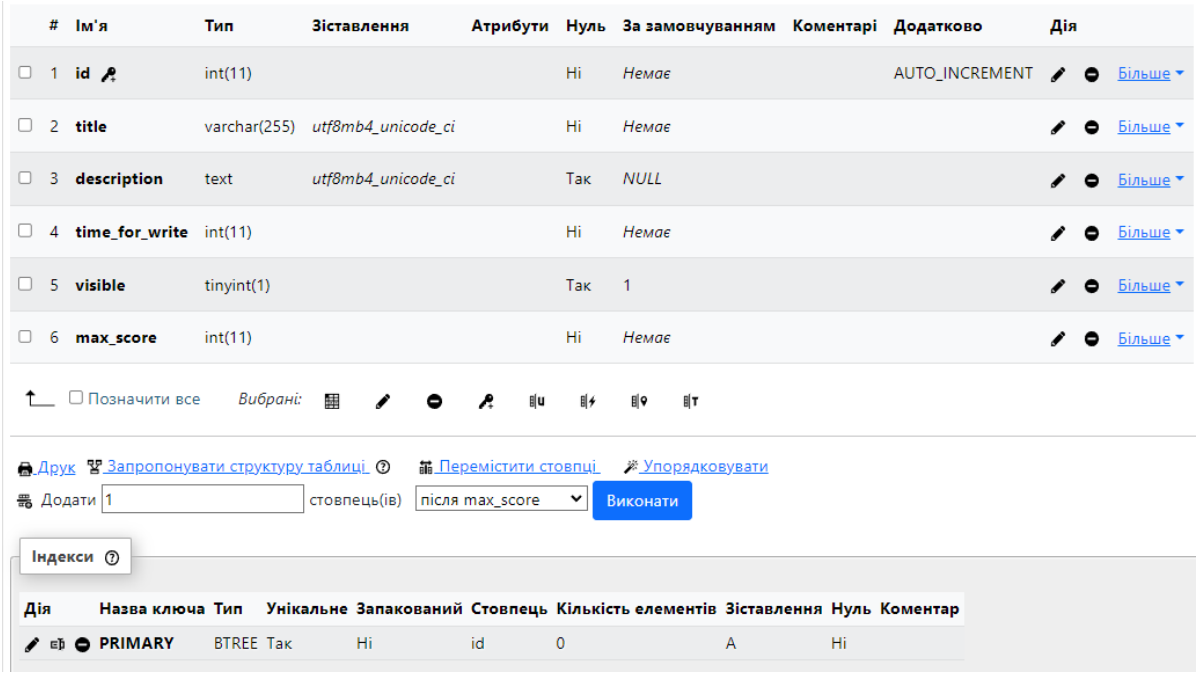


Рисунок 3.3 – Структура таблиці тестів

Таблиця Test_question

Призначення:

Таблиця Test_question зберігає питання, які входять до складу тесту.

Поля:

– id (INT, AUTO_INCREMENT, PRIMARY KEY): Унікальний ідентифікатор питання.

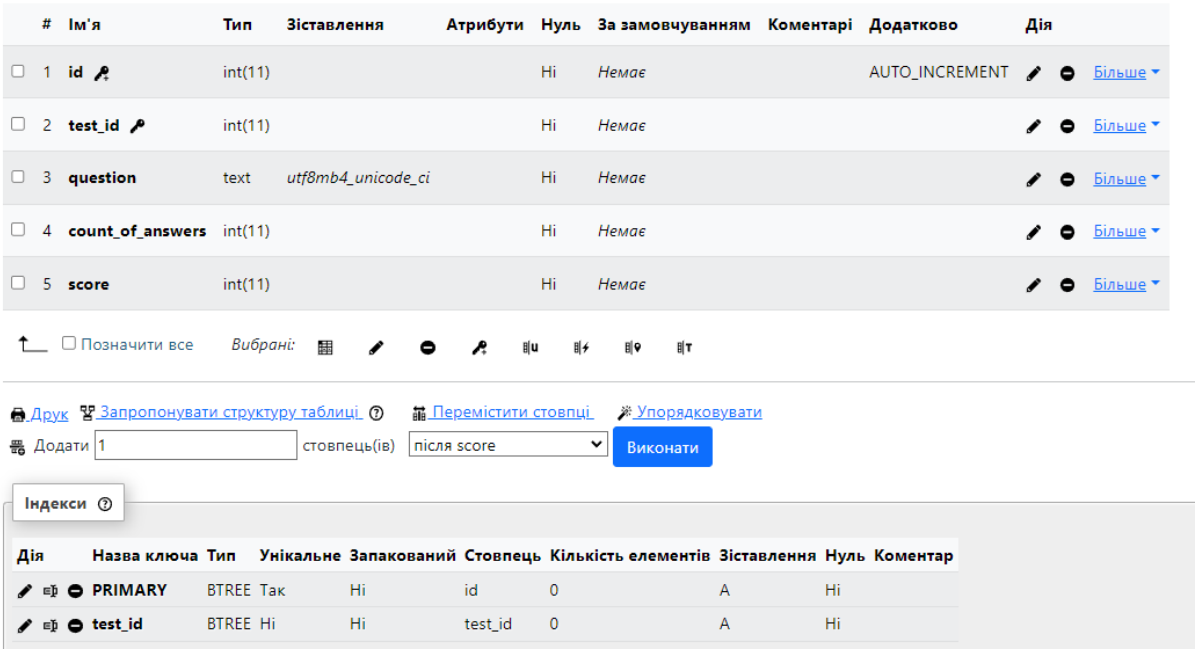
– test_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор тесту, до якого належить це питання. Це поле створює зв’язок з таблицею Test.

– question (TEXT, NOT NULL): Текст питання.

– count_of_answers (INT, NOT NULL): Кількість відповідей, які можуть бути дані на це питання.

– score (INT, NOT NULL): Кількість балів, яка може бути отримана за правильну відповідь на це питання.

Структуру таблиці представлено на рис. 3.4.



#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково	Дія
1	id	int(11)			Ні	*Немає*		AUTO_INCREMENT	Більше
2	test_id	int(11)			Ні	*Немає*			Більше
3	question	text	utf8mb4_unicode_ci		Ні	*Немає*			Більше
4	count_of_answers	int(11)			Ні	*Немає*			Більше
5	score	int(11)			Ні	*Немає*			Більше

Позначити все *Вибрані:*

Дія	Назва ключа	Тип	Унікальне	Запакований	Стовпець	Кількість елементів	Зіставлення	Нуль	Коментар
	PRIMARY	BTREE	Так	Ні	id	0	A	Ні	
	test_id	BTREE	Ні	Ні	test_id	0	A	Ні	

Рисунок 3.4 – Структура таблиці тестових питань

Таблиця Test_answer

Призначення:

Таблиця Test_answer зберігає можливі відповіді на питання тестів.

Поля:

– id (INT, AUTO_INCREMENT, PRIMARY KEY): Унікальний ідентифікатор відповіді.

– question_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор питання, до якого належить ця відповідь. Це поле створює зв’язок з таблицею Test_question.

– answer (TEXT, NOT NULL): Текст відповіді.

– is_correct (BOOLEAN, DEFAULT FALSE): Вказує, чи є ця відповідь правильною.

Структуру таблиці представлено на рис. 3.5.

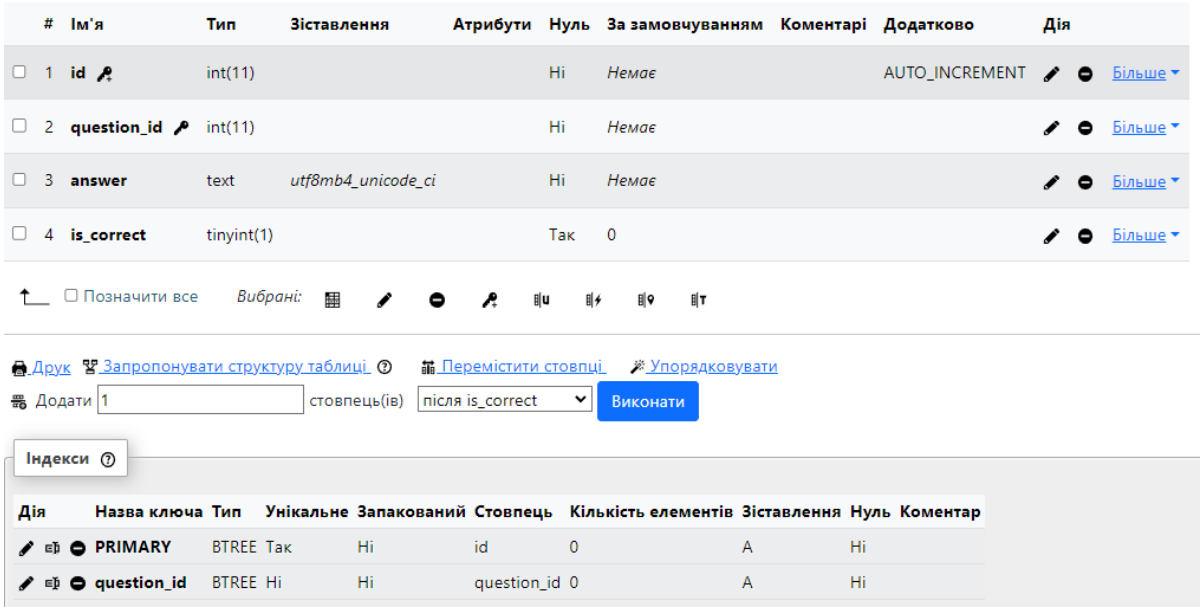


Рисунок 3.5 – Структура таблиці відповідей на тести

Таблиця User_test

Призначення:

Таблиця User_test зберігає результати тестування користувачів.

Поля:

– user_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор користувача, який пройшов тест. Це поле створює зв’язок з таблицею User.

– test_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор тесту, який пройшов користувач. Це поле створює зв’язок з таблицею Test.

– score (INT, NOT NULL): Кількість балів, яку користувач отримав за проходження тесту.

– Особливості:

– PRIMARY KEY складається з user_id та test_id, що гарантує унікальність запису для кожного користувача і тесту.

– Таблиця підтримує цілісність посилань за допомогою зовнішніх ключів, що дозволяє автоматично видаляти результати тестів користувача при видаленні самого користувача або тесту.

Структуру таблиці представлено на рис. 3.6.

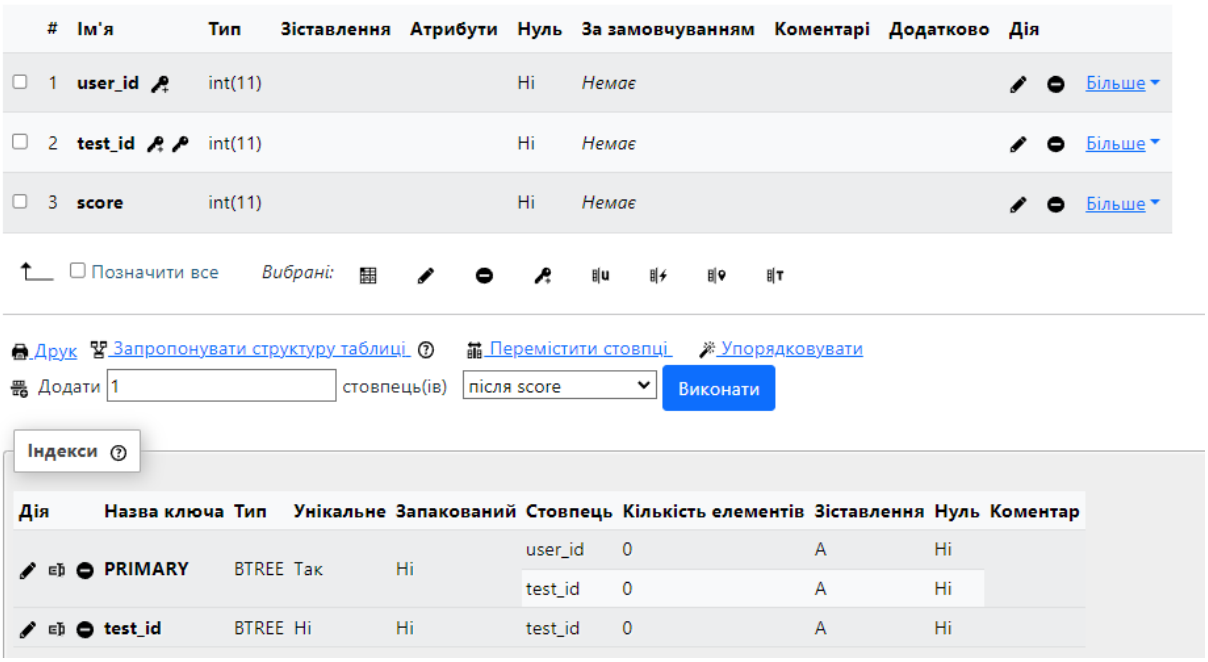


Рисунок 3.6 – Структура таблиці результатів тестів

Таблиця Review:

Призначення:

Таблиця Review зберігає відгуки користувачів про тести, включаючи рейтинг та текст відгуку.

Поля:

– id (INT, AUTO_INCREMENT, PRIMARY KEY): Унікальний ідентифікатор відгуку.

– user_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор користувача, який залишив відгук. Це поле створює зв’язок з таблицею User.

– test_id (INT, NOT NULL, FOREIGN KEY): Ідентифікатор тесту, про який залишено відгук. Це поле створює зв’язок з таблицею Test.

– rating (INT, NOT NULL): Рейтинг тесту, наприклад, від 1 до 5.

– review_text (TEXT): Текст відгуку.

– created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP): Дата і час створення відгуку.

Структуру таблиці відгуків представлено на рис. 3.7

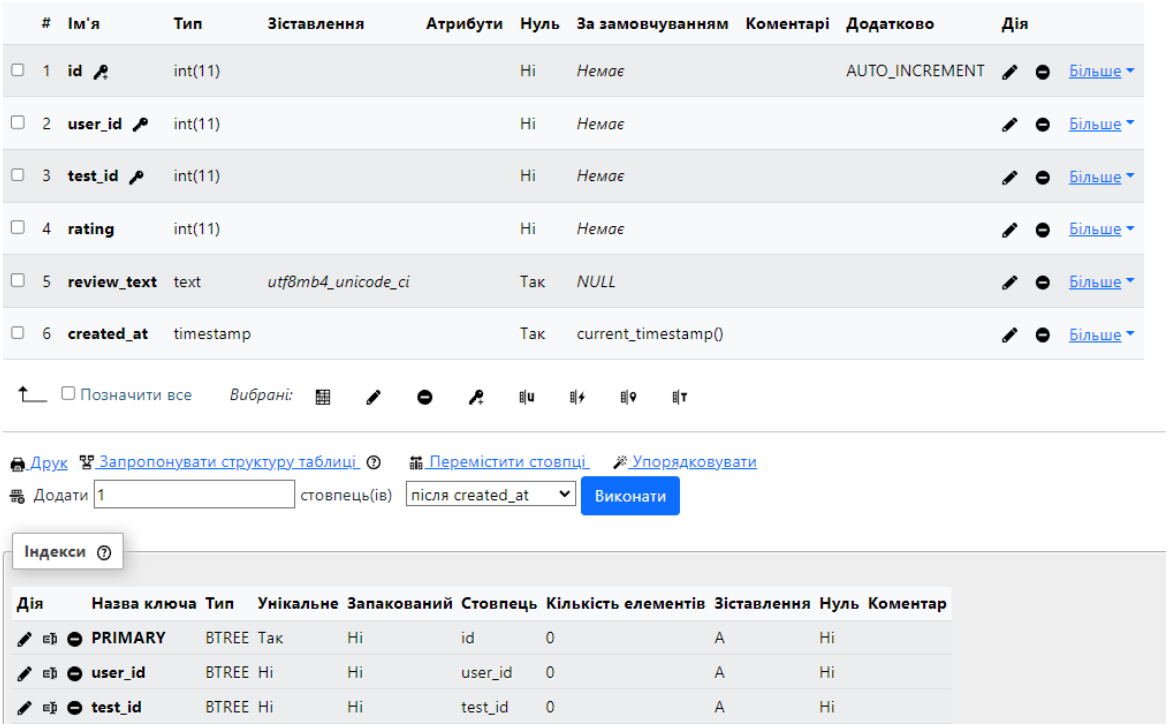


Рисунок 3.7 – Структура таблиці відгуків

Ці таблиці разом утворюють структуровану і логічно зв’язану базу даних, що дозволяє ефективно зберігати, обробляти та отримувати інформацію про користувачів, тести, питання, відповіді та результати тестів.

3.3 Розробка мокапу системи

Мокап системи – це імітація вигляду та функціональності програмного забезпечення, що створюється. Він служить інструментом для візуалізації та концептуалізації інтерфейсу користувача, функцій та потоків роботи системи.

Під час проєктування платформи розроблено декілька мокапів для розробляємої системи.

Мокап головної сторінки, яка відображає коротку інформацію про наявні на платформі тести, відгуки користувачів, аналітику результатів, представлено на рис. 3.8.

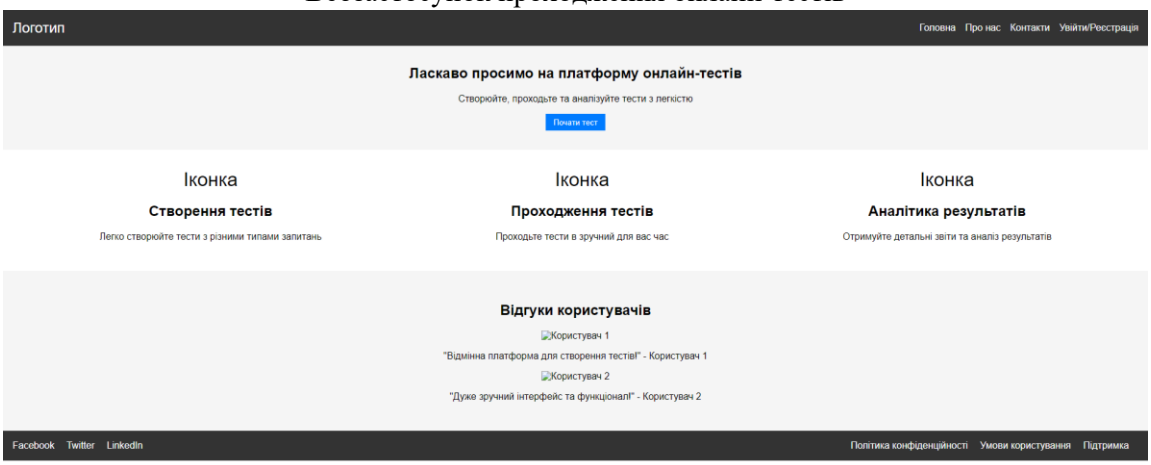


Рисунок 3.8 – Мокап головної сторінки

Мокап сторінки для створення тесту представлено на рис. 3.9.

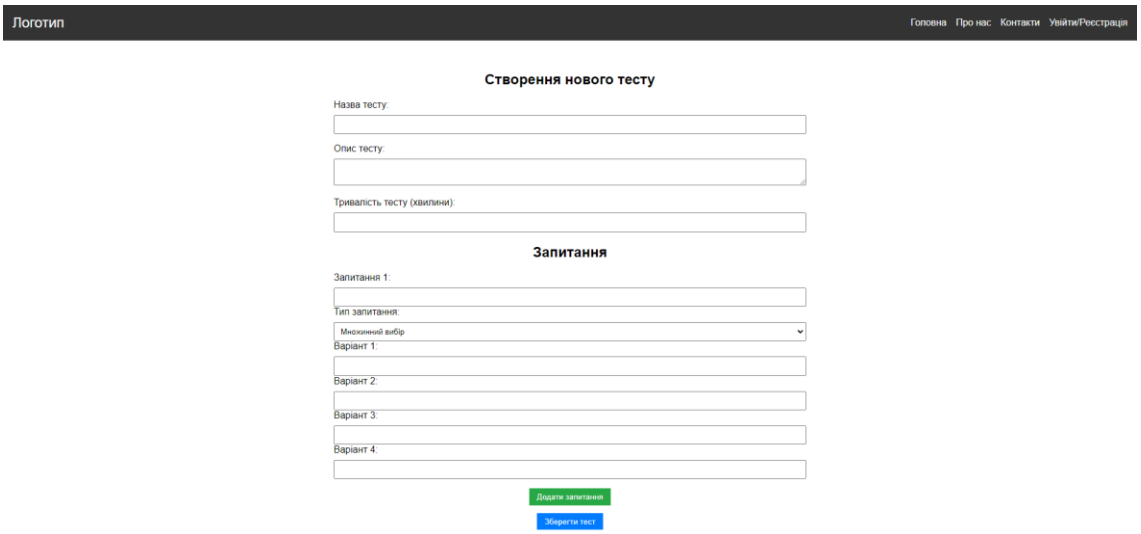


Рисунок 3.9 – Мокап сторінки створення тестів

Мокап сторінки проходження тесту представлено на рис. 3.10.

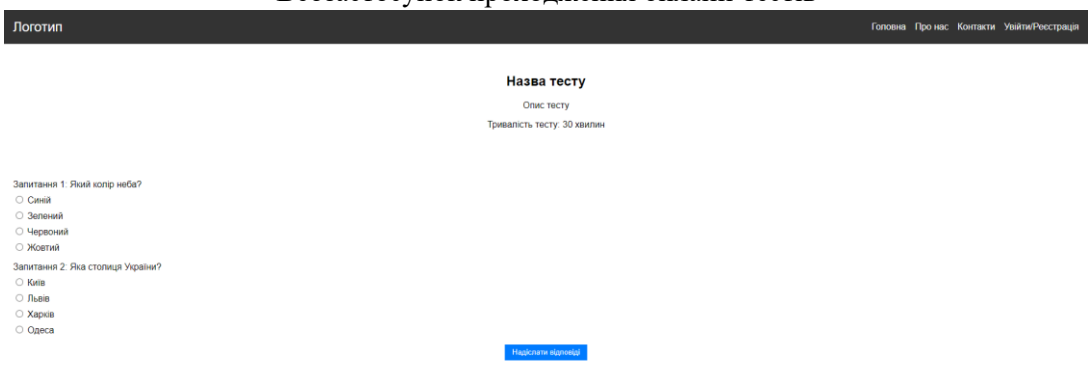


Рисунок 3.10 – Мокап сторінки для проходження тесту

Мокап сторінки результатів тесту представлено на рис. 3.11.

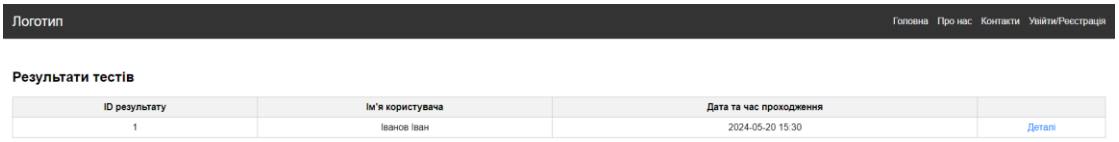


Рисунок 3.11 – Мокап сторінки результатів тесту

Мокап сторінки деталей результату тесту представлено на рис. 3.12.

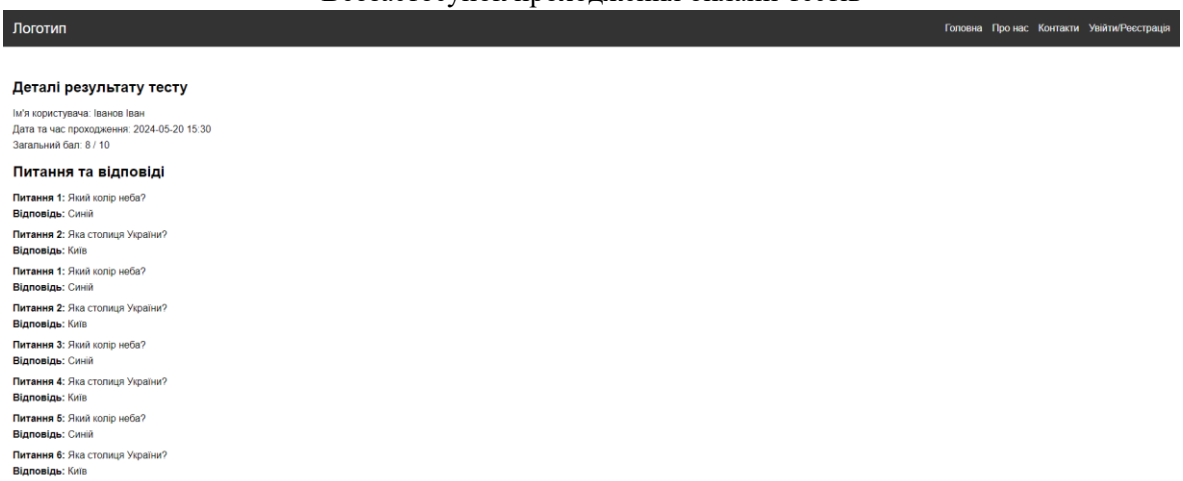


Рисунок 3.12 – Мокап сторінки деталей результату тесту

Даний мокап дозволить більш детально опрацювати функціонал системи.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи описано вибір технологій розробки, а саме:

– MySQL;
– Php;
– Laravel;
– OpenServer.

Представлено модель та структуру бази даних та відображено реалізований мокап вебзастосунку.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Реалізація моделей та контролерів

Під час розробки вебзастосунку розроблено моделі для зв’язку бази даних з вебзастосунком.

Review

Модель Review представляє огляд тесту, який залишає користувач. Включає інформацію про оцінку тесту, текст відгуку та дату створення.

Поля:

– id: унікальний ідентифікатор огляду;

– user_id: ідентифікатор користувача, який залишив огляд (зовнішній ключ до таблиці User);

– test_id: ідентифікатор тесту, до якого відноситься огляд (зовнішній ключ до таблиці Test);

– rating: оцінка тесту, значення від 1 до 5;

– review_text: текст відгуку користувача;

– created_at: дата і час створення огляду.

Код моделі відгуків представлено на рис. 4.1.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Review extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'test_id', 'rating', 'review_text'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function test()
    {
        return $this->belongsTo(Test::class);
    }
}
```

Рисунок 4.1 – Модель відгуків

Test

Модель Test відображає інформацію про тести, доступні для проходження користувачами.

Поля:

– id: унікальний ідентифікатор тесту;

– title: назва тесту;

– description: опис тесту;

– time_for_write: час, що виділений на проходження тесту (у хвилинах);

– visible: прапорець, що вказує на видимість тесту (1 - видимий, 0 - прихований);

– max_score: максимальний бал, який можна отримати за тест.

Модель таблиці тестів представлена на рис. 4.2.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Test extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'description', 'time_for_write', 'visible', 'max_score'];

    public function questions()
    {
        return $this->hasMany(TestQuestion::class);
    }

    public function reviews()
    {
        return $this->hasMany(Review::class);
    }

    public function users()
    {
        return $this->belongsToMany(User::class, 'user_tests')->withPivot('score');
    }
}
```

Рисунок 4.2 – Модель тестів

TestAnswer

Модель TestAnswer представляє відповіді на питання тесту, включаючи інформацію про правильність відповідей.

Поля:

– id: унікальний ідентифікатор відповіді;

– question_id: ідентифікатор питання, до якого відноситься відповідь (зовнішній ключ до таблиці Test_question);

– answer: текст відповіді;

– is_correct: прапорець, що позначає, чи є ця відповідь правильною (1 - правильна, 0 - неправильна).

Модель таблиці відповідей на тести представлена на рис. 4.3.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TestAnswer extends Model
{
    use HasFactory;

    protected $fillable = ['question_id', 'answer', 'is_correct'];

    public function question()
    {
        return $this->belongsTo(TestQuestion::class);
    }
}
```

Рисунок 4.3 – Модель відповідей на тести

TestQuestion

Модель TestQuestion представляє питання, що використовується у тестах, включаючи кількість відповідей та бали, які можна отримати за це питання.

Поля:

– id: унікальний ідентифікатор питання;

– test_id: ідентифікатор тесту, до якого відноситься питання (зовнішній ключ до таблиці Test);

– question: текст питання;

– count_of_answers: кількість можливих відповідей на питання;

– score: бали, які можна отримати за це питання.

Модель для таблиці питань для тестів представлена на рис. 4.4.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TestQuestion extends Model
{
    use HasFactory;

    protected $fillable = ['test_id', 'question', 'count_of_answers', 'score'];

    public function test()
    {
        return $this->belongsTo(Test::class);
    }

    public function answers()
    {
        return $this->hasMany(TestAnswer::class);
    }
}
```

Рисунок 4.4 – Модель питань для тестів

User

Модель User представляє дані користувача, який має доступ до системи тестування.

Поля:

– id: унікальний ідентифікатор користувача;

– username: ім'я користувача;

– email: електронна адреса користувача (унікальна);

– password: хеш пароля користувача.

Модель для таблиці користувачів представлена на рис. 4.5.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = ['username', 'email', 'password'];

    protected $hidden = ['password', 'remember_token'];

    public function reviews()
    {
        return $this->hasMany(Review::class);
    }

    public function tests()
    {
        return $this->belongsToMany(Test::class, 'user_tests')->withPivot('score');
    }
}
```

Рисунок 4.5 – Модель користувачів

UserTest

Модель UserTest зберігає результати тестування користувачів, включаючи отриманий бал за тест.

Поля:

– user_id: ідентифікатор користувача, який проходив тест (зовнішній ключ до таблиці User);

– test_id: ідентифікатор тесту, за який був отриманий результат (зовнішній ключ до таблиці Test);

– score: бал, отриманий користувачем за тест.

Модель для таблиці історії тестів користувача представлено на рис. 4.6.

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserTest extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'test_id', 'score'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function test()
    {
        return $this->belongsTo(Test::class);
    }
}
```

Рисунок 4.6 – Модель аналітики

Ці моделі відображають основні сутності вашої системи тестування і включають необхідні дані для управління та збереження інформації про тести, огляди, користувачів та їх результати.

Після створення моделей, розроблено контролери для керування даними у таблицях бд.

ReviewController

Контролер ReviewController відповідає за обробку дій з оглядами тестів.

Методи**:**

- index: Відображення списку оглядів;
- show: Відображення конкретного огляду;
- create: Відображення форми для створення нового огляду;
- store: Збереження нового огляду у базі даних;
- edit: Відображення форми для редагування існуючого огляду;
- update: Оновлення існуючого огляду у базі даних;
- destroy: Видалення огляду з бази даних.

Відображення коду контролеру представлено на рис. 4.7.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Review;
use Illuminate\Http\Request;

class ReviewController extends Controller
{
    public function index()
    {
        $reviews = Review::all();
        return view('reviews.index', compact('reviews'));
    }

    public function create()
    {
        return view('reviews.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'user_id' => 'required',
            'test_id' => 'required',
            'rating' => 'required|integer|min:1|max:5',
            'review_text' => 'nullable|string',
        ]);

        Review::create($request->all());

        return redirect()->route('reviews.index')->with('success', 'Review created successfully.');
    }

    public function show(Review $review)
    {
        return view('reviews.show', compact('review'));
    }

    public function edit(Review $review)
    {
        return view('reviews.edit', compact('review'));
    }

    public function update(Request $request, Review $review)
    {
        $request->validate([
            'user_id' => 'required',
            'test_id' => 'required',
            'rating' => 'required|integer|min:1|max:5',
            'review_text' => 'nullable|string',
        ]);

        $review->update($request->all());

        return redirect()->route('reviews.index')->with('success', 'Review updated successfully.');
    }

    public function destroy(Review $review)
    {
        $review->delete();

        return redirect()->route('reviews.index')->with('success', 'Review deleted successfully.');
    }
}
```

Рисунок 4.7 – Код відображення контролеру відгуків

TestController

Контролер TestController відповідає за управління даними тестів.

Методи**:**

– index: Відображення списку тестів;

– show: Відображення конкретного тесту;

– create: Відображення форми для створення нового тесту;

– store: Збереження нового тесту у базі даних;
– edit: Відображення форми для редагування існуючого тесту;
– update: Оновлення існуючого тесту у базі даних;
– destroy: Видалення тесту з бази даних.

Відображення коду контролеру представлено на рис. 4.8.

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index()
    {
        $users = User::all();
        return view('users.index', compact('users'));
    }

    public function create()
    {
        return view('users.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'username' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ]);

        $user = User::create([
            'username' => $request->username,
            'email' => $request->email,
            'password' => bcrypt($request->password),
        ]);

        return redirect()->route('users.index')->with('success', 'User created successfully.');
    }

    public function show(User $user)
    {
        return view('users.show', compact('user'));
    }

    public function edit(User $user)
    {
        return view('users.edit', compact('user'));
    }

    public function update(Request $request, User $user)
    {
        $request->validate([
            'username' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,
            'password' => 'nullable|string|min:8|confirmed',
        ]);

        $user->update([
            'username' => $request->username,
            'email' => $request->email,
            'password' => $request->password ? bcrypt($request->password) : $user->password,
        ]);

        return redirect()->route('users.index')->with('success', 'User updated successfully.');
    }

    public function destroy(User $user)
    {
        $user->delete();

        return redirect()->route('users.index')->with('success', 'User deleted successfully.');
    }
}
```

Рисунок 4.8 – Відображення коду контролеру тестів

TestAnswerController

Контролер TestAnswerController керує відповідями на питання тестів.

Методи**:**

– store: Збереження нової відповіді для певного питання;

– update: Оновлення існуючої відповіді для питання;

– destroy: Видалення відповіді з бази даних.

Відображення коду контролеру представлено на рис. 4.9.

```php
<?php
namespace App\Http\Controllers;

use App\Models\TestAnswer;
use Illuminate\Http\Request;

class TestAnswerController extends Controller
{
    public function index()
    {
        $answers = TestAnswer::all();
        return view('test_answers.index', compact('answers'));
    }

    public function create()
    {
        return view('test_answers.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'question_id' => 'required|integer',
            'answer' => 'required|string',
            'is_correct' => 'boolean',
        ]);

        TestAnswer::create($request->all());

        return redirect()->route('test_answers.index')->with('success', 'Answer created
successfully.');
    }

    public function show(TestAnswer $answer)
    {
        return view('test_answers.show', compact('answer'));
    }

    public function edit(TestAnswer $answer)
    {
        return view('test_answers.edit', compact('answer'));
    }

    public function update(Request $request, TestAnswer $answer)
    {
        $request->validate([
            'question_id' => 'required|integer',
            'answer' => 'required|string',
            'is_correct' => 'boolean',
        ]);

        $answer->update($request->all());

        return redirect()->route('test_answers.index')->with('success', 'Answer updated
successfully.');
    }

    public function destroy(TestAnswer $answer)
    {
        $answer->delete();

        return redirect()->route('test_answers.index')->with('success', 'Answer deleted
successfully.');
    }
}
```

Рисунок 4.9 – Відображення коду контролеру відповідей до тестів

TestQuestionController

Контролер TestQuestionController управляє питаннями в тестах.

Методи**:**

– store: Збереження нового питання для певного тесту;

– update: Оновлення існуючого питання;

– destroy: Видалення питання з бази даних.

Відображення коду контролеру представлено на рис. 4.10.

```php
<?php
namespace App\Http\Controllers;

use App\Models\TestQuestion;
use Illuminate\Http\Request;

class TestQuestionController extends Controller
{
    public function index()
    {
        $questions = TestQuestion::all();
        return view('test_questions.index', compact('questions'));
    }

    public function create()
    {
        return view('test_questions.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'test_id' => 'required|integer',
            'question' => 'required|string',
            'count_of_answers' => 'required|integer',
            'score' => 'required|integer',
        ]);

        TestQuestion::create($request->all());

        return redirect()->route('test_questions.index')->with('success', 'Question created
successfully.');
    }

    public function show(TestQuestion $question)
    {
        return view('test_questions.show', compact('question'));
    }

    public function edit(TestQuestion $question)
    {
        return view('test_questions.edit', compact('question'));
    }

    public function update(Request $request, TestQuestion $question)
    {
        $request->validate([
            'test_id' => 'required|integer',
            'question' => 'required|string',
            'count_of_answers' => 'required|integer',
            'score' => 'required|integer',
        ]);

        $question->update($request->all());

        return redirect()->route('test_questions.index')->with('success', 'Question updated
successfully.');
    }

    public function destroy(TestQuestion $question)
    {
        $question->delete();

        return redirect()->route('test_questions.index')->with('success', 'Question deleted
successfully.');
    }
}
```

Рисунок 4.10 – Відображення коду контролеру питань до тестів

UserController

Контролер UserController відповідає за управління користувачами системи.

Методи**:**

– register: Реєстрація нового користувача;

– login: Авторизація користувача в системі;

– logout: Вихід користувача з системи;

– profile: Відображення профілю користувача;

– updateProfile: Оновлення інформації в профілі користувача.

Відображення коду контролеру представлено на рис. 4.11.

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserTest;
use Illuminate\Http\Request;

class UserTestController extends Controller
{
    public function index()
    {
        $userTests = UserTest::all();
        return view('user_tests.index', compact('userTests'));
    }

    public function create()
    {
        return view('user_tests.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'user_id' => 'required|integer',
            'test_id' => 'required|integer',
            'score' => 'required|integer',
        ]);

        UserTest::create($request->all());

        return redirect()->route('user_tests.index')->with('success', 'User Test created
successfully.');
    }

    public function show(UserTest $userTest)
    {
        return view('user_tests.show', compact('userTest'));
    }

    public function edit(UserTest $userTest)
    {
        return view('user_tests.edit', compact('userTest'));
    }

    public function update(Request $request, UserTest $userTest)
    {
        $request->validate([
            'user_id' => 'required|integer',
            'test_id' => 'required|integer',
            'score' => 'required|integer',
        ]);

        $userTest->update($request->all());

        return redirect()->route('user_tests.index')->with('success', 'User Test updated
successfully.');
    }

    public function destroy(UserTest $userTest)
    {
        $userTest->delete();

        return redirect()->route('user_tests.index')->with('success', 'User Test deleted
successfully.');
    }
}
```

Рисунок 4.11 – Відображення коду контролеру користувачів

UserTestController

Контролер UserTestController відповідає за обробку результатів тестування користувачів.

Методи**:**

– store: Збереження результатів тесту для конкретного користувача;

– update: Оновлення результатів тесту для користувача;

– destroy: Видалення результатів тесту з бази даних.

Відображення коду контролеру представлено на рис. 4.12.

```php
<?php
namespace App\Http\Controllers;

use App\Models\Test;
use Illuminate\Http\Request;

class TestController extends Controller
{
    public function index()
    {
        $tests = Test::all();
        return view('tests.index', compact('tests'));
    }

    public function create()
    {
        return view('tests.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'nullable|string',
            'time_for_write' => 'required|integer',
            'visible' => 'boolean',
            'max_score' => 'required|integer',
        ]);

        Test::create($request->all());

        return redirect()->route('tests.index')->with('success', 'Test created successfully.');
    }

    public function show(Test $test)
    {
        return view('tests.show', compact('test'));
    }

    public function edit(Test $test)
    {
        return view('tests.edit', compact('test'));
    }

    public function update(Request $request, Test $test)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'nullable|string',
            'time_for_write' => 'required|integer',
            'visible' => 'boolean',
            'max_score' => 'required|integer',
        ]);

        $test->update($request->all());

        return redirect()->route('tests.index')->with('success', 'Test updated successfully.');
    }

    public function destroy(Test $test)
    {
        $test->delete();

        return redirect()->route('tests.index')->with('success', 'Test deleted successfully.');
    }
}
```

Рисунок 4.12 – Відображення коду контролеру тестів користувача

Ці контролери забезпечують реалізацію всіх основних операцій CRUD (створення, читання, оновлення, видалення) для відповідних моделей у застосунку. Кожен з них відповідає за конкретну сутність в системі тестування, забезпечуючи правильну роботу з даними та їх взаємодію з інтерфейсом користувача.

4.2 Реалізація інтерфейсу вебзастосунку

Реалізація вебсайту є ключовою частиною розробки проєкту, де абстрактні концепції та плани перетворюються у працюючий програмний продукт. Цей розділ детально описує процес створення вебсайту для створення та управління онлайн-тестами, починаючи від вибору технологій і закінчуючи інтеграцією та тестуванням.

Розробка вебсайту включає кілька етапів, кожен з яких має свої особливості та виклики. Важливою частиною цього процесу є вибір технологій, що забезпечують надійність, масштабованість та зручність у використанні. Для проєкту обрано наступні технології:

– MySQL – як система керування базами даних, яка забезпечує надійне зберігання та швидкий доступ до даних.

– PHP – як серверна мова програмування, що дозволяє створювати динамічні вебсторінки та взаємодіяти з базою даних.

– Laravel – як PHP-фреймворк, який спрощує розробку застосунків, надаючи зручний інструментарій для побудови складних вебзастосунків.

– OpenServer – як локальне середовище розробки, що дозволяє налаштувати і запускати сервер, базу даних і вебсайт на локальному комп'ютері.

Головна сторінка є центральним вузлом платформи, що забезпечує користувачам зручний доступ до основних функцій. Вона містить навігаційне меню, що дозволяє швидко переходити до створення тестів, перегляду результатів та інших розділів сайту.

На головній сторінці також розміщено інформаційні блоки, які надають користувачам загальну інформацію про платформу, її можливості та новини.

Реалізація головної сторінки представлена на рис. 4.12.

Рисунок 4.12 – Головна сторінка вебзастосунку

Сторінка для створення тестів надає користувачам зручні інструменти для створення тестів, включаючи додавання питань різних типів, налаштування часу

проходження та складності тесту. Інтерфейс сторінки інтуїтивно зрозумілий, що дозволяє швидко налаштовувати тести без спеціальних технічних знань.

Реалізацію сторінки для створення тестів представлено на рис. 4.13.

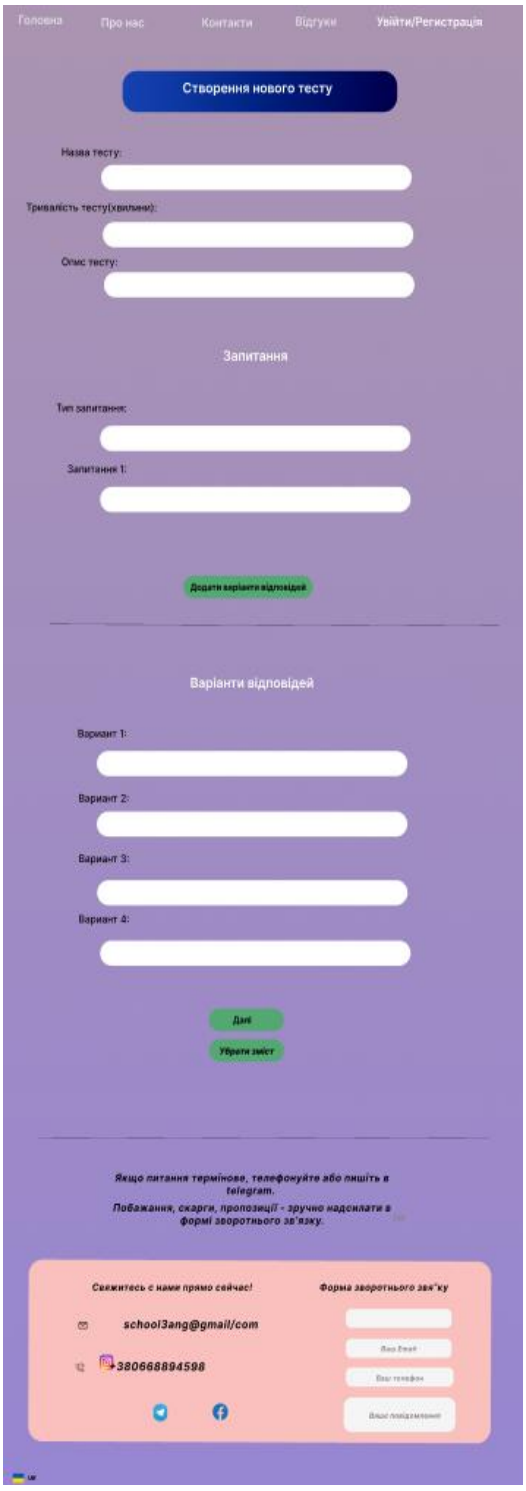


Рисунок 4.13 – Сторінка для створення тестів

Сторінка авторизації забезпечує безпечний вхід користувачів до їхніх облікових записів. На цій сторінці користувачі вводять свої логін та пароль, після

чого отримують доступ до персоналізованих функцій платформи. Для забезпечення безпеки даних, сторінка авторизації використовує сучасні методи шифрування та автентифікації.

Реалізацію сторінки авторизації представлено на рис. 4.14.

Рисунок 4.14 – Сторінка для авторизації

Сторінка реєстрації дозволяє новим користувачам створювати облікові записи на платформі. Користувачам потрібно заповнити форму з основною

інформацією, такою як ім'я, електронна пошта та пароль. Після успішної реєстрації користувачі отримують доступ до всіх можливостей платформи.

Реалізацію сторінки реєстрації представлено на рис. 4.15.

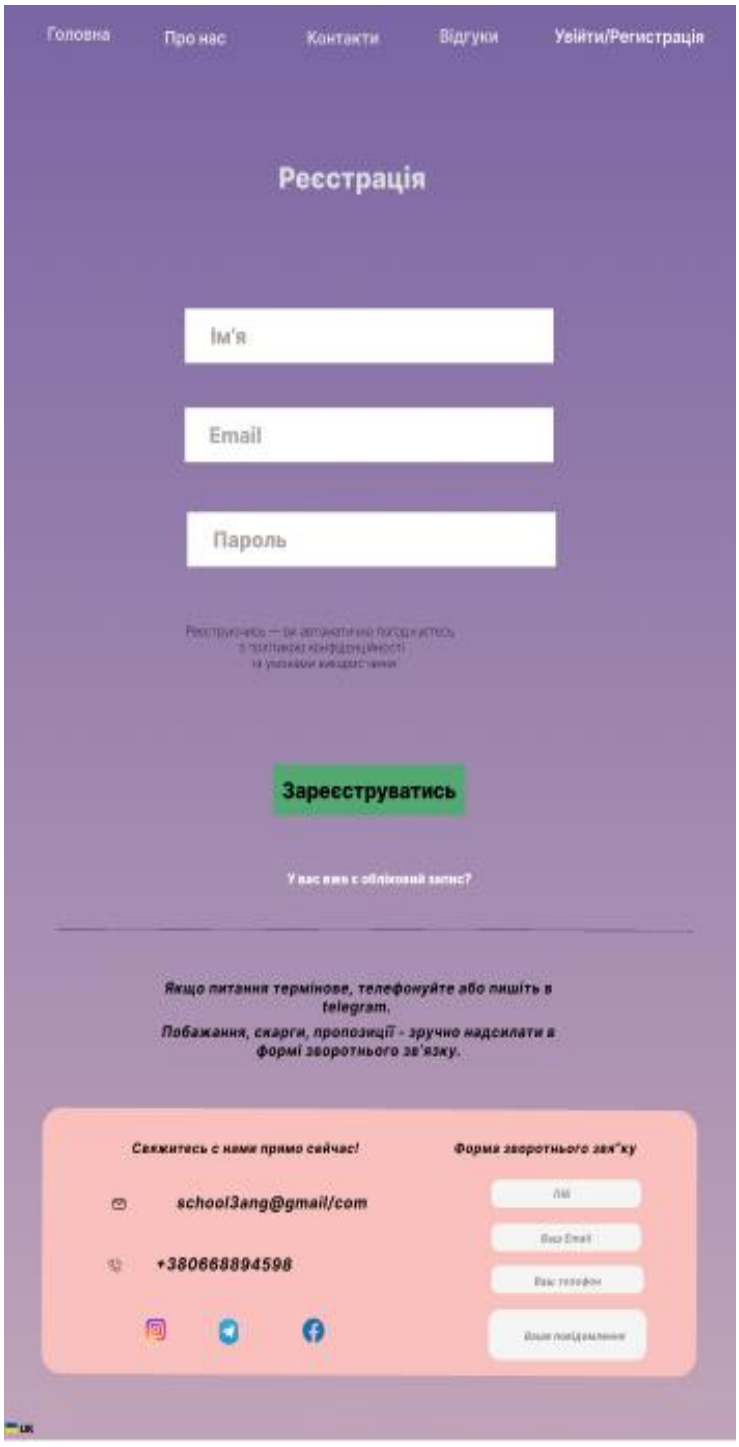


Рисунок 4.15 – Сторінка реєстрації

Сторінка з відгуками дозволяє користувачам залишати свої відгуки про платформу та переглядати думки інших користувачів. Відгуки допомагають покращувати якість послуг платформи та адаптувати її до потреб користувачів.

Інтерфейс сторінки дозволяє легко додавати та сортувати відгуки за різними критеріями.

Реалізацію сторінки з відгуками представлено на рис. 4.16.

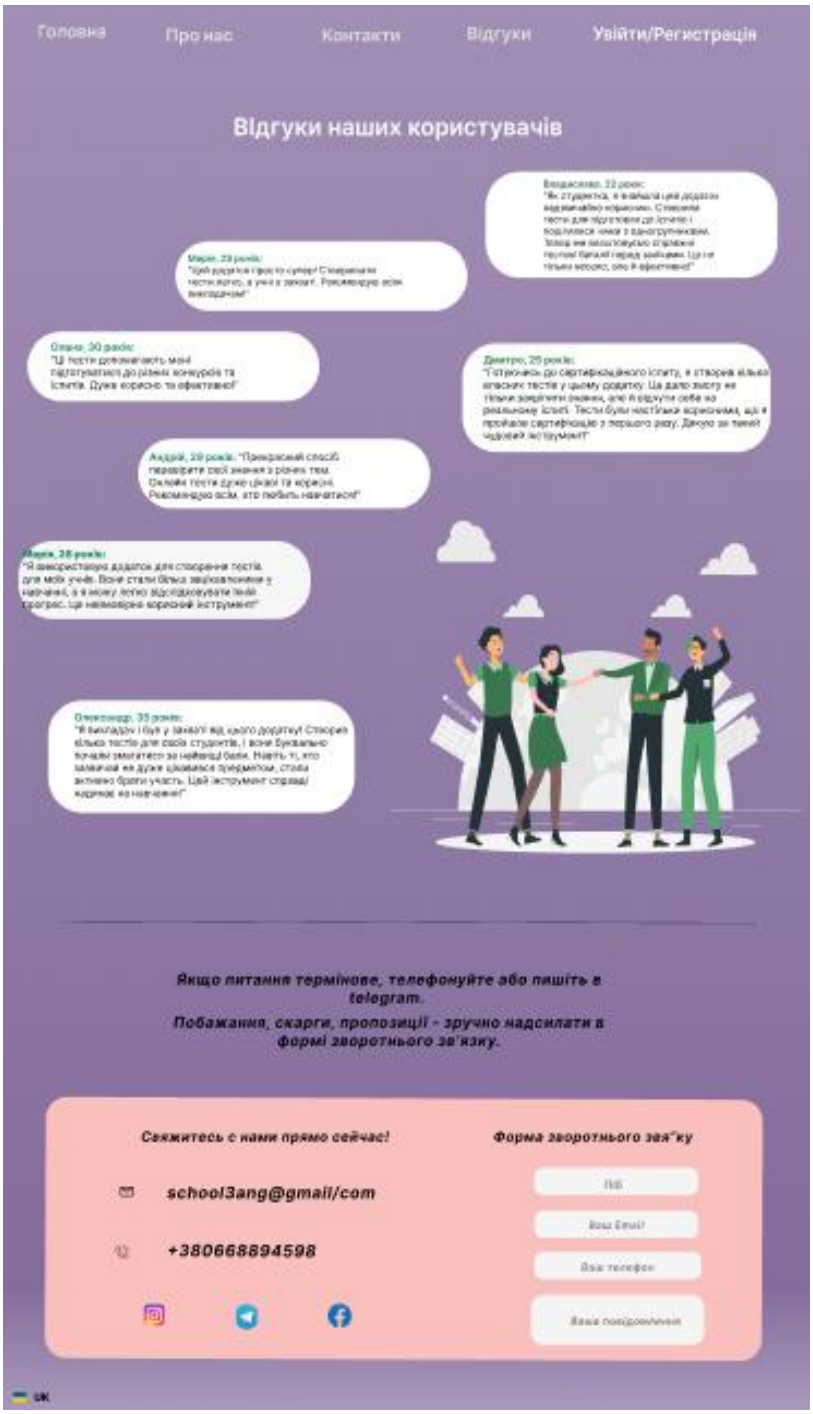


Рисунок 4.16 – Сторінка з відгуками

Сторінка з інформацією про платформу містить детальну інформацію про функціональні можливості платформи, її переваги та особливості. Вона також

включає розділ з часто заданими питаннями (FAQ), що допомагає користувачам швидко знаходити відповіді на найпоширеніші питання.

Реалізацію сторінки яка містить інформацію про платформу представлено на рис. 4.17.

Рисунок 4.17 – Сторінка про платформу

Сторінка з результатами надає користувачам можливість переглядати результати пройдених тестів. Вона містить детальну інформацію про кожен тест,

включаючи правильні та неправильні відповіді, бали та рекомендації для покращення. Інтерфейс сторінки дозволяє зручно аналізувати результати та відслідковувати прогрес у навчанні.

Реалізацію сторінки з результатами представлено на рис. 4.18.

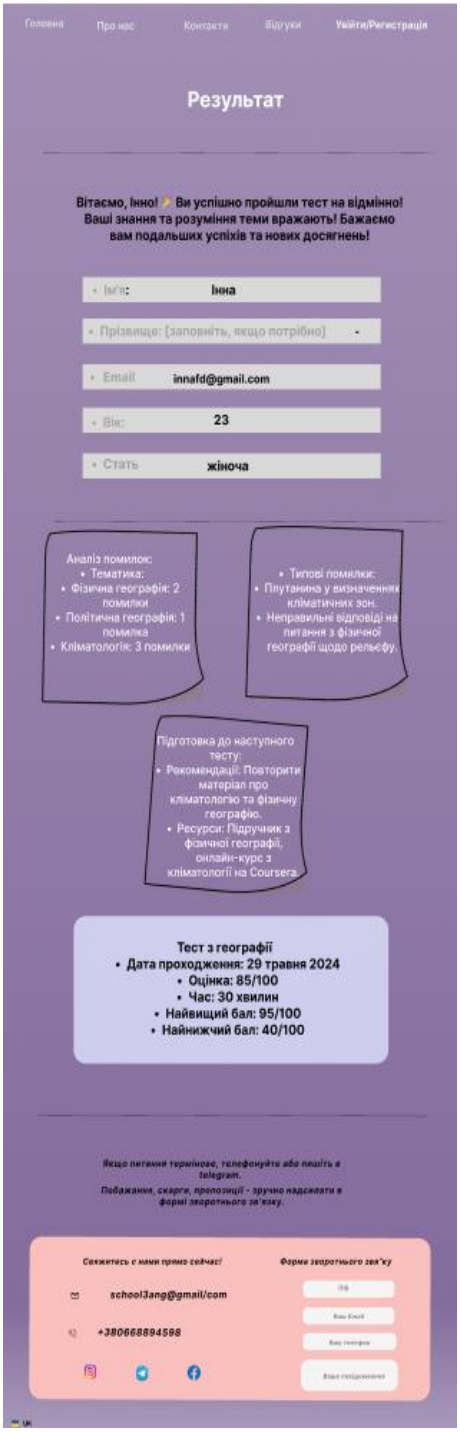


Рисунок 4.18 – Сторінка результатів тесту

Дана сторінка відображає статистику користувача по тестам, а саме його результати та відсоток правильних відповідей.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи представлено реалізацію вебзастосунку для створення онлайн-тестів.

Відображено реалізацію наступних сторінок:

- головна сторінка;
- сторінка авторизації;
- сторінка реєстрації;
- сторінка створення тестів;
- сторінка відгуків.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи успішно досягнуто всіх поставлених завдань.

Розроблено функціональну та зручну платформу для створення онлайн-тестів, яка забезпечує інтуїтивний підхід до створення різноманітних тестів. Платформа дозволяє користувачам легко створювати тести та керувати ними, полегшуючи навчальний процес для вчителів та учнів.

Спроєктовано базу даних для зберігання та обробки інформації про тести, запитання та результати. Створена структура бази даних забезпечує ефективне зберігання та швидкий доступ до необхідної інформації, що сприяє безперебійній роботі платформи.

Оптимізовано швидкодію платформи та адаптовано її для різних пристроїв та екранів. Завдяки цьому користувачі можуть зручно працювати з платформою на будь-якому пристрої, включаючи комп’ютери, планшети та смартфони, що забезпечує доступність та зручність використання.

Забезпечено можливість авторизації та реєстрації користувачів, що дозволяє персоналізувати роботу з платформою, зберігаючи дані користувачів та їхні результати тестування. Це підвищує безпеку та конфіденційність інформації.

Реалізовано функцію автоматичної перевірки відповідей та генерацію аналітичних звітів. Це дозволяє швидко оцінювати результати тестування та надавати детальні звіти, що сприяє ефективному аналізу та підвищенню якості навчального процесу.

Таким чином, у процесі виконання кваліфікаційної роботи створено повнофункціональний вебзастосунок, який відповідає всім поставленим завданням, забезпечуючи зручність, швидкодію та адаптивність для різних пристроїв та екранів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Google forms: вебсайт. URL: https://www.google.com/intl/ru_ua/forms/about/ (Last accessed: 03.04.2024)

2. Moodle: вебсайт. URL: https://www.google.com/intl/ru_ua/forms/about/ (Last accessed: 03.04.2024)

3. Kahoot: вебсайт. URL: https://www.google.com/intl/ru_ua/forms/about/ (Last accessed: 03.04.2024)

4. Мова програмування Php: підручник . Welling L., Thomson L. PHP and MySQL Web development. Sams publishing, 2003 p. C. 365.

5. Мова програмування Php: підручник . Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. " O'Reilly Media, Inc.", 2014 p. C. 522.

6. Мова програмування Php: підручник . Nixon R. Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites. " O'Reilly Media, Inc.", 2012 p. C. 637.

7. Ullman L. PHP and MySQL for dynamic Web sites: visual quickpro guide. Peachpit Press, 2011 p. C. 569.

8. Yuliano T. Pengenalan Php //IlmuKomputer. com. 2007 p. C. 474.

9. Mitchell W. H. PHP. 2008 p. C. 543.

10. Stauffer M. Laravel: Up & Running. " O'Reilly Media, Inc.", 2023 p. C. 278.

11. Bean M. Laravel 5 essentials. Packt Publishing Ltd, 2015 p. C. 15.

12. Фреймворк Laravel: підручник. Armel J. Web application development with Laravel PHP Framework version 4. 2014 p. C. 213

13. Фреймворк Laravel: підручник. McCool S. Laravel starter. Packt Publishing, 2012 p. C. 253.

14. Фреймворк Laravel: підручник. Sinha S., Dave H. J. Beginning Laravel Packt Publishing, 2017 p. C. 125.

15. Фреймворк Laravel: підручник. He R. Y. Design and implementation of web based on Laravel framework //2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014). Atlantis Press, 2015. С. 304.

16. Фреймворк Laravel: підручник. Yadav N., Rajpoot D. S., Dhakad S. K. LARAVEL: a PHP framework for e-commerce website //2019 Fifth International Conference on Image Information Processing (ICIIP). IEEE, 2019. С. 508.

17. Фреймворк Laravel: підручник. Dangar H. Learning laravel 4 application development. Packt Publishing, 2013 p. С. 165.

18. Мова програмування Sql: підручник . DuBois P. MySQL. Addison-Wesley, 2013 p. С. 64.

19. Мова програмування Sql: підручник . Гольцман В. MySQL 5.0. 2018 р. С. 115.

20. Мова програмування Php: підручник . Welling L., Thomson L. PHP and MySQL Web development. Sams publishing, 2003 p. С. 235.

21. Php docs: вебсайт. URL: https://www.php.net/docs.php (Last accessed: 03.04.2024)

22. Laravel docs: вебсайт. URL: https://laravel.com/docs/11.x/readme (Last accessed: 03.04.2024)

23. MySQL docs: вебсайт. URL: https://dev.mysql.com/doc/ (Last accessed: 03.04.2024)

ДОДАТОК А – КОД РОЗРОБКИ

```
<?php
namespace App\Http\Controllers;

use App\Models\Test;
use Illuminate\Http\Request;

class TestController extends Controller
{
    public function index()
    {
        $tests = Test::all();
        return view('tests.index', compact('tests'));
    }

    public function create()
    {
        return view('tests.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'nullable|string',
            'time_for_write' => 'required|integer',
            'visible' => 'boolean',
            'max_score' => 'required|integer',
        ]);

        Test::create($request->all());

        return redirect()->route('tests.index')->with('success', 'Test created successfully.');
```

```
        }

        public function show(Test $test)
        {
            return view('tests.show', compact('test'));
        }

        public function edit(Test $test)
        {
            return view('tests.edit', compact('test'));
        }

        public function update(Request $request, Test $test)
        {
            $request->validate([
                'title' => 'required|string|max:255',
                'description' => 'nullable|string',
                'time_for_write' => 'required|integer',
                'visible' => 'boolean',
                'max_score' => 'required|integer',
            ]);

            $test->update($request->all());

            return                redirect()->route('tests.index')-
>with('success', 'Test updated successfully.');
        }

        public function destroy(Test $test)
        {
            $test->delete();

            return                redirect()->route('tests.index')-
>with('success', 'Test deleted successfully.');
        }
    }
```

```
<?php
namespace App\Http\Controllers;

use App\Models\TestAnswer;
use Illuminate\Http\Request;

class TestAnswerController extends Controller
{
    public function index()
    {
        $answers = TestAnswer::all();
        return view('test_answers.index', compact('answers'));
    }

    public function create()
    {
        return view('test_answers.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'question_id' => 'required|integer',
            'answer' => 'required|string',
            'is_correct' => 'boolean',
        ]);

        TestAnswer::create($request->all());

        return redirect()->route('test_answers.index')->with('success', 'Answer created successfully.');
    }

    public function show(TestAnswer $answer)
    {
```

```
            return view('test_answers.show', compact('answer'));
        }

        public function edit(TestAnswer $answer)
        {
            return view('test_answers.edit', compact('answer'));
        }

        public function update(Request $request, TestAnswer $answer)
        {
            $request->validate([
                'question_id' => 'required|integer',
                'answer' => 'required|string',
                'is_correct' => 'boolean',
            ]);

            $answer->update($request->all());

            return redirect()->route('test_answers.index')->with('success', 'Answer updated successfully.');
        }

        public function destroy(TestAnswer $answer)
        {
            $answer->delete();

            return redirect()->route('test_answers.index')->with('success', 'Answer deleted successfully.');
        }
    }
    <?php
    namespace App\Http\Controllers;

    use App\Models\TestQuestion;
    use Illuminate\Http\Request;
```

```
class TestQuestionController extends Controller
{
    public function index()
    {
        $questions = TestQuestion::all();
        return                    view('test_questions.index',
compact('questions'));
    }

    public function create()
    {
        return view('test_questions.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'test_id' => 'required|integer',
            'question' => 'required|string',
            'count_of_answers' => 'required|integer',
            'score' => 'required|integer',
        ]);

        TestQuestion::create($request->all());

        return      redirect()->route('test_questions.index')-
>with('success', 'Question created successfully.');
    }

    public function show(TestQuestion $question)
    {
        return                     view('test_questions.show',
compact('question'));
    }

    public function edit(TestQuestion $question)
```

```
        {
            return                          view('test_questions.edit',
compact('question'));
        }

        public  function  update(Request  $request,  TestQuestion
$question)
        {
            $request->validate([
                'test_id' => 'required|integer',
                'question' => 'required|string',
                'count_of_answers' => 'required|integer',
                'score' => 'required|integer',
            ]);

            $question->update($request->all());

            return       redirect()->route('test_questions.index')-
>with('success', 'Question updated successfully.');
        }

        public function destroy(TestQuestion $question)
        {
            $question->delete();

            return       redirect()->route('test_questions.index')-
>with('success', 'Question deleted successfully.');
        }
    }

    <?php

    namespace App\Http\Controllers;

    use App\Models\Review;
    use Illuminate\Http\Request;
```

```
class ReviewController extends Controller
{
    public function index()
    {
        $reviews = Review::all();
        return view('reviews.index', compact('reviews'));
    }

    public function create()
    {
        return view('reviews.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'user_id' => 'required',
            'test_id' => 'required',
            'rating' => 'required|integer|min:1|max:5',
            'review_text' => 'nullable|string',
        ]);

        Review::create($request->all());

        return redirect()->route('reviews.index')->with('success', 'Review created successfully.');
    }

    public function show(Review $review)
    {
        return view('reviews.show', compact('review'));
    }

    public function edit(Review $review)
    {
```

```
            return view('reviews.edit', compact('review'));
        }

        public function update(Request $request, Review $review)
        {
            $request->validate([
                'user_id' => 'required',
                'test_id' => 'required',
                'rating' => 'required|integer|min:1|max:5',
                'review_text' => 'nullable|string',
            ]);
            $review->update($request->all());

            return              redirect()->route('reviews.index')-
>with('success', 'Review updated successfully.');
        }
        public function destroy(Review $review)
        {
            $review->delete();

            return              redirect()->route('reviews.index')-
>with('success', 'Review deleted successfully.');
        }
    }
```